Filipe Alexandre Bandeira Cerqueira

Licenciado em Engenharia Informática

Um Sistema Publicador/Subscritor com Persistência de Dados para Redes de Dispositivos Móveis

Dissertação para obtenção do Grau de Mestre em

Engenharia Informática

Orientador: Hervé Miguel Cordeiro Paulino, Prof. Auxiliar, Universidade Nova de Lisboa

Júri

Presidente: Carla Maria Gonçalves Ferreira
Vogais: Eduardo Resende Brandão Marques
Hervé Miguel Cordeiro Paulino

FACULDADE DE CIÊNCIAS E TECNOLOGIA
UNIVERSIDADE NOVA DE LISBOA

Setembro, 2017

Um Sistema Publicador/Subscritor com Persistência de Dados para Redes de Dispositivos Móveis

Este documento foi gerado utilizando o processador (pdf)LaTeX, com base no template "novathesis" [1] desenvolvido no Dep. Informática da FCT-NOVA [2].
[1] https://github.com/joaomlourenco/novathesis [2] http://www.di.fct.unl.pt

Aos meus pais e irmão.

Agradecimentos

Muitos são os agradecimentos que têm de ser feitos.

Agradeço vivamente ao meu orientador, Hervé Paulino, pela orientação, conselhos, apoio e dedicação prestada a esta dissertação, não podendo deixar de fazer um agradecimento especial para o João A. Silva pela disponibilidade, ajuda e esforço, estando sempre disponível para uma troca de ideias e para uma explicação em dúvidas existentes. Agradeço também a todos os participantes do projeto "Hyrax: Crowd-Sourcing Mobile Devices to Develop Edge Clouds" [57], todos eles foram importantes para o sucesso desta dissertação. Um agradecimento para o Departamento de Ciência de Computadores da FCUP, pela disponibilidade apresentada na minha deslocação para efetuar a avaliação experimental, em múltiplos dispositivos móveis. Agradeço aos professores Fernando Silva, Luís Lopes e Eduardo Marques, por tão bem me terem recebido, e ao João Rodrigues por todo o apoio disponibilizado nesse período.

Agradeço ao departamento de Informática da FCT-UNL e a todos os seus docentes por todo o conhecimento que me transmitiram.

Como não pode deixar de ser, um muito obrigado aos meus companheiros nesta "batalha", António Relvas, Sérgio Pimentel, Everton Nascimento, Bruno Firmino e Alexandra Silva, e um agradecimento especial para o André Carrusca por todo o apoio ao longo desta dissertação, mas também por tudo aquilo que passamos nestes 5 anos.

À minha família! Aos meus pais que sempre acreditaram em mim, deixando-me tomar o percurso que decidi, foram fundamentais nesta dissertação e na minha vida. Ao meu irmão, pois, sei que sou um exemplo para ele. Agradeço a todos aqueles que me apoiaram, amigos e família, permitindo terminar esta fase da minha vida. Com uma palavra especial para os meus padrinhos pelo apoio e ajuda prestados.

Por fim, e sendo deixada para o fim por um motivo especial, o meu maior agradecimento para a Ana Lúcia Sousa. Pela sua constante presença, paciência, otimismos nos momentos mais difíceis, pelos desabafos, por nunca me deixar desistir e por estar sempre ao meu lado.

Obrigado a quem partiu.

Um grande obrigado a todos!

Resumo

Os dispositivos móveis são atualmente uma das principais fontes de geração de conteúdos pessoais. A portabilidade e a crescente capacidade destes dispositivos permitem que os mesmos sejam utilizados nas mais variadas atividades, lúdicas e não só. Tal tendência tem sido acompanhada pelo progressivo interesse em partilhar (em tempo real) os conteúdos gerados. No entanto, a comum utilização de serviços centralizados para suportar a partilha de dados entre utilizadores requer sistemas de servidores com grande capacidade de resposta e coloca muita carga na rede, reduzindo a sua largura de banda com dados que (muitas vezes) podem ser partilhados na periferia.

Nesta dissertação apresentamos uma implementação em Android do Thyme, um sistema publicador/subscritor com persistência de dados para redes de dispositivos móveis, que fornece um mecanismo de disseminação e armazenamento de dados adaptado para este tipo de redes, sem requerer acesso à Internet. No Thyme, o tempo é uma dimensão de primeira ordem, associando um intervalo de tempo a cada subscrição, em que o início e fim do intervalo podem ser referentes ao futuro, presente ou passado.

Como caso de uso apresentamos uma aplicação Android para partilha de fotografias.

Os resultados obtidos mostram, que o trabalho desenvolvido apresenta o comportamento desejado num ambiente composto por um número considerável de dispositivos, conseguindo inclusivamente lidar com mobilidade e *churn*. Foi possível ainda verificar que o Thyme apresenta tempos de resposta que garantem a interatividade com o utilizador, comprovando poder ser utilizado em eventos de curta e media duração, com um baixo consumo de energia, permitindo que a aplicação seja utilizada sem gastos excessivos de bateria.

Palavras-chave: Dispositivos móveis, armazenamento distribuído, redes móveis, publicador/subscritor, computação na periferia, Android.

Abstract

Mobile devices are currently one of the main sources of user-generated content. The portability and increasing capacity of these devices allow them to be used in various activities. This tendency has been accompanied by the progressive interest in sharing (in real time) the generated content. However, the common use of centralized services to support the sharing of data between users requires highly responsive services and places a high load on the network, reducing its bandwidth with data that (often) can be shared in the network edge.

In this thesis we apply Thyme to real-world networks of Android devices, a data persistent publish/subscribe system for wireless networks of mobile devices, which provides a mechanism for data dissemination and storage adapted for this type of network, without requiring Internet access. In Thyme, time is a first order dimension, associating a time interval with each subscription, where the start and end of the interval may refer to the future, present, or past.

As a use case we present an Android application for photo sharing.

The experimental results show that developed work presents the desired behavior in an environment composed by a considerable number of devices, even managing to deal with mobility and churn. It was also possible to verify that Thyme presents response times that guarantee interactivity with the user, proved to be able to be used in events of short and medium duration, with a low energy consumption, allowing the application to be used without excessive battery costs .

Keywords: Mobile devices, distributed storage, publish/subscribe, mobile networks, edge Computing, Android.

Índice

Lista de Figuras

Lista de Tabelas

Listagens

Capítulo 1

Introdução

Esta dissertação apresenta uma implementação em Android do Thyme, um sistema publicador/subscritor com persistência de dados que permite a disseminação e armazenamento de dados em redes de dispositivos móveis. Neste capítulo apresenta-se a motivação para o desenvolvimento deste trabalho, qual o problema que pretendemos resolver e qual a solução proposta para a resolução do mesmo. Posteriormente descrevem-se as contribuições e publicações que resultaram desta dissertação.

1.1 Motivação

Nos últimos anos, o número de dispositivos móveis, como smartphones e tablets, tem vindo a aumentar na sociedade a um ritmo bastante elevado, tornando-se cada vez mais omnipresentes. De acordo com uma previsão efetuada pela Cisco [16], é esperado que o número de dispositivos móveis exceda os 11.6 mil milhões no ano de 2021, representando um aumento de 52.6% comparativamente ao ano de 2015, onde este número não ultrapassava os 7.6 mil milhões. Estes dados permitem perceber que o domínio da computação está a mudar, deixando de ser cada vez menos fixo, tornando-se cada vez mais móvel.

O crescimento da utilização destes dispositivos está relacionado com a evolução das suas características. Atualmente é possível encontrar dispositivos móveis com grande poder de computação, elevado espaço de armazenamento, assim como boa capacidade de comunicação, podendo em alguns casos ser comparados a computadores pessoais. O mercado atual oferece dispositivos móveis com processadores octa-core, com memórias RAM de 6GB e superiores, com armazenamento interno de 128GB, módulos de comunicação como Wi-Fi 802.11 [2], Bluetooth 5.0 [76] e NFC [29], como por exemplo o OnePlus 5 [58]. Estes dispositivos também têm adquirido recursos muito apreciados pelos utilizadores,

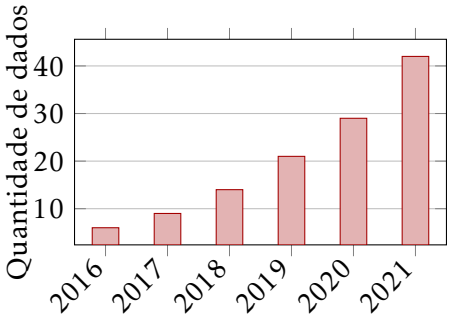


Figura 1.1: Quantidade de dados transferidos na rede via smartphones, em milhões de TB por mês [16].

tornando-se essenciais [73], como câmara fotográfica/vídeo de elevada qualidade (16MP) ou os mecanismos de localização cada vez mais confiáveis como o sistema GPS.

O aumento da utilização dos dispositivos móveis aliado à sua evolução, leva a que estes se tornem uma das principais formas de acesso à Internet e uma das principais fontes de conteúdo criado pelos utilizadores [16]. Este crescimento na criação de conteúdo por parte do utilizador, como fotografias, vídeos e documentos, leva também ao crescimento da disseminação e partilha desses conteúdos com outros utilizadores. De acordo com valores publicados em [16], prevê-se que em 2021 a partilha de dados gerados pelo utilizador, utilizando a rede móvel seja de 1102867 TB por mês, sendo que em 2016 esse valor era de 151874 TB por mês. Estes valores tornam-se ainda mais interessantes quando se verifica que a quantidade de dados transferidos na rede usando *smartphones* subiu cerca de 58% desde 2016 até 2017 e irá apresentar uma subida superior a 600% até ao ano de 2021 (Figura 1.1).

Quando falamos de disseminação e partilha de dados entre dispositivos móveis, grande parte dos utilizadores utilizam serviços hospedados em *clouds* para a realização destas tarefas, denominando-se este paradigma de *mobile cloud computing*. Estes serviços hospedados em *clouds* centralizadas, conhecidas vulgarmente apenas como *cloud*, apresentam serviços de armazenamento e computação, hospedados em servidores de grandes empresas como Google, Amazon ou Microsoft. Utilizando um serviço remoto de terceiros (Figura 1.2a), é possível armazenar documentos, fotografias, músicas, entre outros dados; partilhar esses dados com outros utilizadores e ainda fazer o processamento desses dados [5, 6, 36].

São cada vez mais os utilizadores de dispositivos móveis que utilizam serviços de *cloud*, devido ao aumento do número destes dispositivos e das necessidades dos utilizadores. Apesar da sua utilização, estes sistemas apresentam problemas quando utilizados em contextos móveis. O facto de ser necessário o envio dos dados para a *cloud* só por sí já é um problema quando se fala em dispositivos móveis. O envio dos dados para a *cloud* obriga a ligação à Internet sendo necessária uma infraestrutura de comunicação configurada para o efeito ou uma ligação de dados móveis (3G). Em ambos os casos a latência da

operação de envio e receção dos dados, quando necessário, pode ser bastante alta, devido: ao número de utilizadores, relativamente próximos, a usufruir da mesma rede que pode ser elevado, diminuindo a largura de banda e aumentando assim a latência; a ligação à Internet pode ser de baixa velocidade, comum nas redes de dados móveis; o tamanho dos dados podem ser elevados sendo demoroso a sua transferência pela rede. Este aumento da latência do envio e receção dos dados e a própria necessidade desse envio e receção podem causar transtorno ao utilizador, pois além de se poder tornar uma tarefa bastante frustrante e desagradável, o consumo de energia pode levar à falha do dispositivo. Além disso, não se pode esquecer que à ligação a Internet pode não ser possível, devido à falta de infraestrutura que forneça ligação à Internet ou pode apresentar custos elevados devido ao consumo dos dados móveis, o que torna a comunicação com a *cloud* pouco acessível.

Com a evolução das características e funcionalidades dos dispositivos móveis, os utilizadores têm vindo a tornar-se mais exigentes na realização deste tipo de tarefas, procurando constantemente uma melhor opção para as suas necessidades. De forma a tentar colmatar estas necessidades surgiram novos paradigmas computacionais como *mobile edge computing* [37] ou *mobile edge cloud* [20]. Estes paradigmas *edge cloud* alavancam os recursos de um conjunto de dispositivos móveis colocados em proximidade, formando uma nuvem na periferia, como na Figura 1.2b, de forma a melhorar a experiência do utilizador. Estes permitem efetuar disseminação e partilha dos dados gerados, garantem poder de processamento e armazenamento mais próximo dos utilizadores, permitindo uma menor latência no acesso aos conteúdos, um melhor comportamento da rede, não existe dependência de ligação à Internet e têm como objetivo melhorar a experiência do utilizador [11, 38, 50].

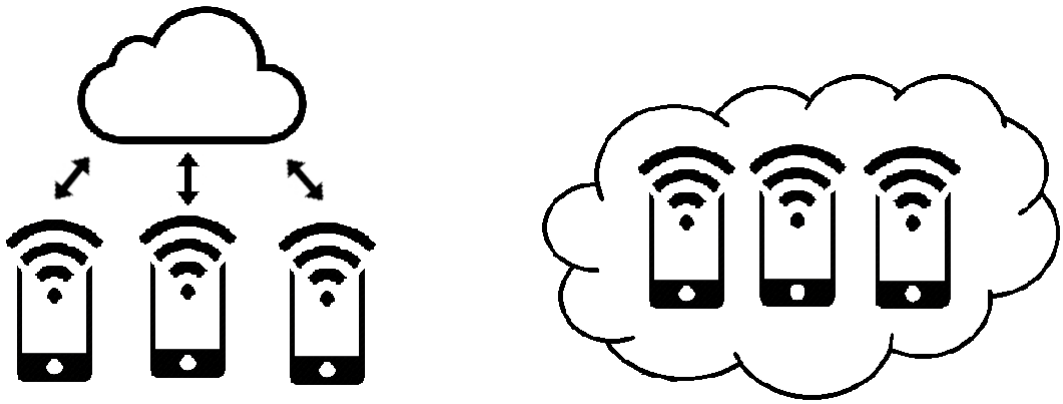
Comparando as duas soluções, ambas apresentam vantagens e desvantagens em diferentes cenários, como é possível observar na Tabela 1.1. Analisando os dois paradigmas, pode-se concluir que o paradigma *mobile cloud computing* é a melhor solução quando pretendemos partilhar conteúdos em diferentes locais do globo ou quando pretendemos computar/armazenar grandes quantidades de dados. Por outro lado, o paradigma *edge cloud* é mais indicado quando o acesso à Internet é deficiente ou inexistente e quando se pretende partilhar e armazenar dados entre dispositivos geograficamente próximos.

O paradigma *edge cloud* é bastante favorável em eventos esporádicos, nomeadamente festas de anos e reuniões; em locais onde não é possível utilizar uma infraestrutura de comunicação como em cenários de catástrofes ou locais desérticos; e ainda em locais onde a ligação à Internet não apresenta capacidade para lidar com as necessidades dos utilizadores devido à elevada carga na rede ou ao tráfego gerado ser muito elevado, como em eventos desportivos ou concertos.

A utilização deste paradigma é o mais indicado para estes cenários pois evita o envio de conteúdos para locais remotos; diminui o tempo de latência e carga na rede; a sua configuração é rápida e barata e além disso apresenta um consumo de bateria relativamente baixo para os dispositivos em questão [20].

Tabela 1.1: Diferenças entre *Cloud Computing* e *Edge Cloud* [74].

Características	***Cloud Computing***	***Edge Cloud***
Implementação[1]	Centralizada	Distribuída
Capacidade computacional e de armazenamento	Ampla	Limitada
Infraestrutura	Fixa	Altamente dinâmica
Estabilidade dos elementos	Estáticos	Dinâmicos
Conectividade	Estável	Irregular
Custos de instalação	Elevados	Reduzidos
Tempo de configuração	Alto	Baixo
Planeamento	Antecipado	Oportunista
Durabilidade	Definitiva	Temporária
Jitter	Alto	Baixo
Distância ao utilizador	Distante	Próxima
Latência	Alta	Baixa



a *Mobile cloud computing* b *Mobile Edge Computing*

Figura 1.2: Comparação entre *Mobile Cloud Computing* e *Mobile Edge Computing*.

1.2 Problema

O problema que esta dissertação visa resolver foca-se nos problemas de latência e conectividade que os utilizadores enfrentam quando pretendem partilhar e armazenar de forma persistente os dados através de dispositivos móveis, colocados geograficamente próximos. Tendo em conta as características dos dispositivos assim como a proximidade dos dispositivos móveis, será vantajoso partilhar e armazenar os dados na periferia (nos próprios dispositivos móveis).

As lacunas que pretendemos colmatar encontram-se nos mais diversos cenários, como por exemplo em eventos esporádicos, festas de anos, eventos desportivos ou reuniões.

[1] *Em inglês a denominação é deployment*.

Nestes eventos é comum estarem presentes grandes quantidades de pessoas, numa zona de proximidade, e estes utilizarem o seu dispositivo móvel para capturar fotografias ou vídeos [24] do evento, gerando grandes quantidades de dados. Nestas situações é comum os utilizadores desejarem partilhar esses conteúdos com outros participantes [10, 24], em tempo real. Apesar de frequente, a disseminação e partilha destes conteúdos entre participantes nem sempre é trivial.

A solução mais comum para a disseminação de dados entre utilizadores em eventos esporádicos é a utilização de serviços centralizados, como a Dropbox [22] ou o Google Drive [35]. No entanto esta solução apresenta problemas na sua utilização. Além dos problemas de conectividade e latência referidos na Secção 1.1, que adicionaria lentidão ou até falha no *upload* e *download* dos dados, seria necessário partilhar com todos os utilizadores interessados a localização dos dados de forma a ser possível efetuar o *upload* e posterior *download* dos dados. Esta tarefa será cada vez mais árdua conforme o aumento do número de utilizadores.

A questão da comunicação dispositivo-a-dispositivo tem sido um foco de investigação por alguns autores tiram partido dos mecanismos de comunicação inerentes aos dispositivos móveis, como o Bluetooth ou Wi-Fi Direct, para estabelecer comunicação entre os dispositivos móveis dispostos numa zona de proximidade [59, 72, 82]. Estes mecanismos de comunicação inerentes aos dispositivos móveis também podem ser utilizados em combinação com uma infraestrutura Wi-Fi ou mesmo um ponto de acesso móvel [80]. Apesar do seu potencial, estes mecanismos não apresentam abstrações suficientemente indicadas para a utilização por parte dos programadores, levando à necessidade de resolução de problemas relacionados com as características do ambiente móvel e da comunicação sem fios.

Utilizando os mecanismos de comunicação referidos anteriormente, foram desenvolvidas aplicações que permitem a troca de dados entre dispositivos geograficamente próximos, como o SuperBeam [48], Xender [3], MediaBowl [67] ou FrostWire [32], mas estes não garantem persistência dos dados, são restritos a um número limitado de recetores e apresentam uma acoplação direta entre o produtor e o consumidor dos dados, o que torna a tarefa de disseminação bastante trabalhosa e maçuda, pois o número de utilizadores é elevado.

Existem trabalhos de investigação sobre o assunto, que propõem sistemas para resolver os problemas descritos, como o Krowd [21], Ephesus [79], iTrust [49] ou MobiTribe [85]. Estes sistemas permitem a partilha e armazenamento de dados em redes de dispositivos móveis mas apresentam lacunas que queremos colmatar, como a não persistência dos dados, não resistência à mobilidade dos nós, o elevado número de mensagens na rede, o elevado tamanho das mensagens, a acoplação entre o produtor e o consumidor e os problemas de escalabilidade da rede.

1.3 Solução Proposta

Esta dissertação enquadra-se no projeto *Hyrax: Crowd-Sourcing Mobile Devices to Develop Edge Clouds* [57] que propõem uma visão inovadora da utilização de nuvens na periferia, utilizando as características dos dispositivos móveis, de forma a permitir a computação, armazenamento e disseminação de dados mais próximos do utilizador. A ideia principal deste projeto é efetuar as tarefas pretendidas utilizando uma nuvem na periferia, isto é, os dispositivos móveis em proximidade, evitando a comunicação ou envio de dados para o exterior, diminuindo assim o tempo de resposta das tarefas em questão, melhorando a experiência do utilizador.

O objetivo do projeto é desenvolver um conjunto de *middleware* completo, isto é, desde da conectividade, passando pela rede, incluindo diversos serviços – onde nos incluímos – mas também o desenvolvimento de aplicações que comprovem o funcionamento dos sistemas, Figura 1.3. Desta forma, participam várias equipas neste projeto sendo que cada equipa contem âmbitos de investigação diferentes, participando em camadas distintas do *middleware*.

Esta dissertação debruça-se sobre a camada de "Serviços" (realçada a laranja na Figura 1.3), onde se pretende oferecer serviço de partilha e armazenamento dos dados partilhados. Neste contexto foi proposto o Thyme [78], um sistema publicador/subscritor com persistência de dados, que se encontra atualmente implementado, usando a linguagem de programação *C++*, para redes *ad-hoc* e avaliado no simulador ns-3 [70].

Nesta dissertação temos como objetivo materializar o modelo Thyme na plataforma Android em redes de dispositivos móveis, para que o Thyme possa ser testado e utilizado em dispositivos e contextos reais.

O modelo Thyme apresenta um sistema publicador/subscritor com persistência de dados que permite disseminação e armazenamento em redes de dispositivos móveis. O Thyme utiliza os recursos dos dispositivos móveis para permitir a disseminação, armazenamento e persistência dos dados sem a necessidade de ligação à Internet.

O modelo Thyme utiliza o paradigma publicador/subscritor utilizando *tags* como

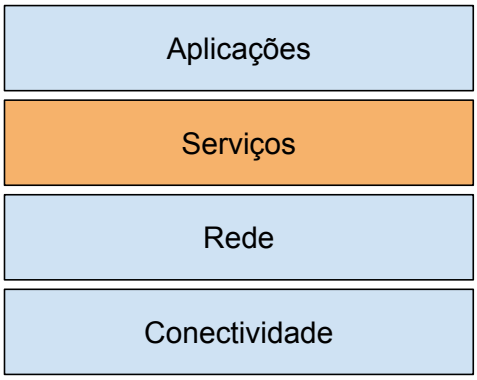


Figura 1.3: Camadas de *middleware*.

identificadores dos objetos para os utilizadores, sendo que as *tags* são palavras, normalmente, relacionadas com o objeto (similar às *hashtags* usadas nas redes sociais). Com este paradigma os utilizadores podem subscrever uma ou várias *tags* e, posteriormente, receber notificações quando um novo objeto é publicado e corresponder às *tags* subscritas. Este paradigma oferece um desacoplamento completo em termos de tempo, espaço e sincronização entre publicadores e subscritores [26] , o que permite interações espontâneas e desacopuladas.

O Thyme diferencia-se dos sistemas publicador/subscritor padrão pois permite efetuar subscrições no passado, de forma a obter dados que foram publicados anteriormente às subscrições, e não apenas no futuro como o habitual [26]. Esta característica é possível porque o tempo é uma dimensão de primeira ordem pois, adicionando ao publicador/subscritor um mecanismo que permite aceder aos dados que foram anteriormente publicados no sistema. Dando um exemplo, numa festa de anos, o utilizador pode apenas estar interessado nos dados publicados depois do jantar e relacionados apenas com a *tag* "diversão". Assim, quando há uma subscrição, o utilizador deverá indicar qual a *tag* que pretende subscrever e especificar qual o intervalo de tempo no qual deseja que a sua subscrição esteja ativa, podendo abranger o presente, o futuro ou o passado. Que seja do nosso conhecimento, o Thyme é o primeiro sistema no contexto de computação móvel, que utiliza paradigma publicador/subscritor onde é possível efetuar subscrições num intervalo de tempo que pode incidir também no passado.

O modelo Thyme pode ser materializado de variadas formas, sendo que uma das abordagens, propostas no artigo [78], utiliza uma solução de armazenamento centrada nos dados que assenta sobre uma tabela de dispersão geográfica (TDG) [68] baseada em células. Visto que a posição geográfica dos dispositivos influencia a topologia da rede sem fios, a nossa abordagem passa por dividir o espaço geográfico em células tratando todos os dispositivos de cada célula como um nó virtual, permitindo assim o armazenamento dos dados por célula. Esta abordagem permite ter uma topologia consciente da rede e além disso permite o acesso aos dados pela sua localização geográfica.

Como caso de estudo, foi desenvolvida uma aplicação para dispositivos móveis com sistema operativo Android [34], onde utilizamos fotografias como conteúdo a ser partilhado e armazenado. Esta aplicação consiste numa galeria de fotografias onde é possível efetuar publicação de fotografias, associando-lhe uma ou várias *tags*, subscrever *tags* e descarregar as fotografias que foram publicadas por outros utilizadores, sendo que o subscritor irá receber uma notificação indicando que, pelo menos uma das *tags* associadas a essa fotografia corresponde a uma das *tags* subscritas por si, podendo assim descarregar essa fotografia para o seu dispositivo.

Resumindo, apresentamos uma implementação do Thyme para dispositivos Android utilizando redes de dispositivos móveis. O Thyme é um sistema publicador/subscritor com persistência de dados que permite disseminação e armazenamento em redes de dispositivos móveis, utilizando o paradigma publicador/subscritor como forma de disseminação de dados, onde o tempo é uma dimensão de primeira ordem permitindo

a subscrição no presente, no futuro e no passado, e uma tabela de dispersão geográfica baseada em células como substrato de armazenamento, que garante persistência dos dados e diminuição do número e tamanho das mensagens na rede. Pretende-se garantir interatividade e satisfação para os utilizadores, excluindo a necessidade de Internet e usufruindo das características dos próprios dispositivos móveis, permitindo assim diminuir a sobrecarga na rede, aumentando a largura de banda e diminuindo a latência na rede.

1.4 Contribuições

Esta dissertação apresenta as seguintes contribuição:

1. Materialização do Thyme, um sistema publicador/subscritor com persistência de dados que permite partilhar e armazenar dados em redes de dispositivos móveis, para dispositivos móveis Android;

2. Uma aplicação para dispositivos móveis Android desenvolvida sobre o Thyme, onde utilizamos fotografias como conteúdo a ser partilhado e armazenado sobre uma rede formada por este tipo de dispositivos móveis;

3. Avaliação experimental do sistema e da aplicação desenvolvida, em ambiente real e simulado.

1.5 Publicações

Na sequência do trabalho realizado nesta dissertação, foram publicados três artigos:

1. Filipe Cerqueira, João A. Silva, João M. Lourenço, e Hervé Paulino. "Um Sistema Publicador/Subscritor com Persistência de Dados para Redes de Dispositivos Móveis", em INForum - Simpósio de Informática. Neste artigo é apresentado o sistema Thyme, a aplicação para Android como caso de uso do sistema e uma avaliação experimental do sistema e da aplicação.

2. Pedro Sanches, António Teófilo, Filipe Cerqueira, João A. Silva, e Hervé Paulino. "Computação distribuída em redes formadas por dispositivos móveis", em INForum - Simpósio de Informática. Neste artigo o sistema apresentado nesta dissertação é utilizado para permitir a computação de dados em redes de dispositivos móveis.

3. Filipe Cerqueira, João A. Silva, João M. Lourenço, e Hervé Paulino. "Towards a Persistent Publish/Subscribe System for Networks of Mobile Devices", em "The 2nd Workshop on Middleware for Edge Clouds & Cloudlets". Neste artigo foi apresentado o Thyme, a materialização que efetuamos em dispositivos móveis Android, a aplicação desenvolvida como caso de estudo do sistema e ainda uma avaliação experimental do sistema e da aplicação.

1.6 Estrutura do documento

Este documento encontra-se dividido em seis capítulos:

Introdução (1) Neste capítulo, o atual, apresenta-se o trabalho desenvolvido. Inicialmente apresenta-se a motivação para este trabalho e qual o problema que se pretende resolver. Posteriormente é abordada a nossa solução e apresenta-se as contribuições desta dissertação e as publicações que o trabalho desta dissertação deu origem.

Estado da arte (2) É importante conhecer os conteúdos mais relevantes para este trabalho de forma a permitir uma melhor compreensão do sistema apresentado e deste documento. Por essa razão no Capítulo 2 apresentam-se os trabalhos e conceitos mais relevantes para esta dissertação. Inicia-se por apresentar o conceito e as características das redes de dispositivos móveis, prossegue-se com os vários sistemas relevantes - a nível de investigação e em lojas de aplicações - sobre armazenamento e disseminação de dados em redes de dispositivos móveis. Em seguida é abordado o paradigma publicador/subscritor - o seu conceito, o seu comportamento em ambientes móveis e em relação a subscrições de dados no passado. A terminar o capítulo é explorada a descoberta e localização dos dados no contexto do nosso trabalho e termina-se o capítulo com um resumo dedicado a conclusões e observações retiradas ao longo do mesmo.

Thyme (3) Este capítulo é dedicado ao sistema que desenvolvemos, abordando o modelo Thyme e como este foi implementado de forma a funcionar em dispositivos móveis, utilizando uma rede formada por este tipo de dispositivos. Ao longo do capítulo são apresentados os tópicos que permitem a compreensão do sistema e as opções que foram tomadas para que este modelo fosse implementado no contexto que pretendemos. O capítulo inicia-se com uma abordagem ao modelo do Thyme, com a Secção 3.1, e segue-se com a Secção 3.2 onde é abordada a arquitetura do sistema implementado, sendo definidos os módulos utilizados assim como as suas interações. No decorrer do capítulo é possível compreender o modelo publicador/subscritor cujas características permitem a obtenção de dados no passado, como os dados são armazenados e encaminhados no sistema, como é feita a gestão da grelha da tabela de dispersão geográfica baseada em células e do espaço geográfico onde o sistema vai funcionar, e como o sistema reage ao movimento dos nós. A terminar o capítulo são apresentados os detalhes de implementação que permitiram implementar o Thyme neste contexto e termina com uma secção sumária.

Galeria de Fotos: caso de estudo (4) Neste capítulo é apresentado o protótipo da aplicação desenvolvida como caso de estudo da implementação do Thyme para dispositivos Android. O capítulo inicia-se com a Secção 4.1 onde é possível obter uma visão geral da aplicação, onde é explicado o objetivo da aplicação, algumas características

relevantes, o seu funcionamento em geral e as operações disponíveis. Na Secção 4.2 são apresentadas as configurações necessárias para inicializar a aplicação tornando-a funcional e qual a razão para tais configurações. Nas restantes secções, de 4.3 até 4.4.6, são apresentadas as operações que podem ser efetuadas pela aplicação e como é possível efetua-las.

Avaliação (5) Neste capítulo é apresentada uma avaliação da implementação do THYME e da aplicação desenvolvida em relação ao seu comportamento, características e funcionalidades. O capítulo inicia com uma descrição das metodologias avaliadas, Secção 5.1, e posteriormente são apresentados os resultados experimentais em dois ambientes distintos: ambiente real (Secção 5.2) e ambiente simulado (Secção 5.3). Nestas secções é possível analisar e obter conclusões do resultados obtidos em ambos os ambientes experimentais. A Secção 5.4 termina o capítulo, onde são apresentadas algumas conclusões e informações relevantes sobre o capítulo.

Conclusão (6) Por fim, o último capítulo deste documento apresenta uma conclusão geral sobre os pontos mais relevantes desta dissertação e na Secção 6.2 são apresentadas algumas direções que podem ser seguidas num trabalho futuro.

Capítulo 2

Estado da arte

Neste capítulo são expostos os trabalhos e conceitos relacionados com o tema desta dissertação, de forma a permitir uma melhor compreensão do sistema apresentado neste documento. Começa-se por discutir, na Secção 2.1, o tema das redes de dispositivos móveis, onde é apresentado o seu conceito e como podem ser construídas e suportadas. Seguidamente, a Secção 2.2 é direcionada para o armazenamento de dados no contexto desta dissertação, onde abordamos algumas características importantes e como estas podem ser garantidas. Na Secção 2.3 apresentam-se os trabalhos que mais se relacionam com o tema desta dissertação, a nível de investigação (Subsecção 2.3.1) e aplicações em lojas de aplicações (Subsecção 2.3.3), abordando as suas características, propriedades e funcionalidades, terminando-se a secção com uma análise crítica comparativa ao nosso sistema (Secção 2.3.2). A Secção 2.4 é dedicada ao paradigma publicador/subscritor, onde é apresentado o conceito do mesmo, como este se aplica atualmente ao contexto móvel (Subsecção 2.4.1) e a subscrição de dados no passado (Subsecção 2.4.2). A terminar, a Secção 2.5 debruça-se sobre os problemas relativos à descoberta e localização dos dados em redes de dispositivos móveis, discutindo as opções existentes e as suas características, onde são abordados os algoritmos de encaminhamento (Subsecção 2.5.1) e posteriormente as alternativas para a localização dos dados (Subsecção 2.5.2). Por fim, na Secção 2.6 apresenta-se um resumo, retirando algumas conclusões sobre o capítulo em questão.

2.1 Redes de dispositivos móveis

As redes de dispositivos móveis, são como o nome indica, constituídas por dispositivos móveis que utilizam mecanismos de comunicação sem fios para comunicar entre si [8, 44]. Estas redes são oportunistas, pois podem ser criadas a qualquer momento e de forma dinâmica pelos dispositivos móveis, utilizando tecnologias de comunicação sem fios, sem a necessidade de infraestruturas ou existência de uma arquitetura centralizada. Neste tipo de redes, os nós têm total mobilidade, podendo mover-se aleatoriamente e organizar-se de várias formas distintas, tornando assim a topologia da rede volátil e imprevisível. Estas redes podem funcionar com conexão à Internet ou apenas de forma autónoma.

Sendo uma rede totalmente descentralizada é necessário saber qual o caminho que um dado pacote tem de percorrer do nó de origem até ao nó de destino. A descoberta deste caminho tem o nome de encaminhamento, *"routing"* em inglês. Como estas redes são constituídas por dispositivos móveis que podem constantemente mover-se, a topologia da rede vai sendo alterada, obrigando assim a que esses caminhos se alterem com bastante frequência, complicando assim o trabalho dos algoritmos de encaminhamento mais comuns, sendo necessária a utilização de protocolos mais adaptáveis às modificações da rede.

Apresentando ligações sem fios, estes tipos de redes expõem alguns problemas relacionados com a capacidade limitada das suas ligações e a largura de banda que este tipo de ligações proporciona.

O contexto envolvendo dispositivos móveis podem ainda trazer o problema da limitação de energia, pois estes dispositivos são alimentados por baterias e sabe-se que estas não são ilimitadas, muito pelo contrário [12]. Além disso, é importante ter em atenção que é normal que os dispositivos móveis tenham características de *hardware* diferentes, sendo que é necessário lidar com essas diferenças entre dispositivos, como por exemplo diferentes alcances e velocidades de transmissão do Wi-Fi ou tempos de vida de baterias diferentes.

Estas redes oportunistas são bastante utilizadas quando é necessário criar uma rede temporária pois, não necessitam de infraestrutura, são redes que são de rápida criação, não necessitam de uma configuração muito complexa nem de administração constante [1].

Segundo as necessidades do nosso sistema e do ambiente que o engloba, a utilização de uma rede oportunista de dispositivos móveis onde existe simetria entre os nós, isto é, não existirem nós com diferentes responsabilidades, seria a solução mais indicada, mas infelizmente não é possível a sua criação e manutenção em dispositivos Android. Existem trabalhos como [33] e [86] onde os autores tentam resolver esses problemas, mas onde é necessário ter permissões de superutilizador *(root)* do dispositivo, funcionalidade que não está acessível nos dispositivos aquando da sua aquisição.

Pretendemos que a utilização da nossa solução não necessite da obrigatoriedade de efetuar *root* ao dispositivo. Pretendemos que a sua utilização seja de fácil acesso: ir a uma loja de aplicações (PlayStore, Apple Store, ou outra), descarregar a aplicação, instalar

no dispositivo e ser possível a sua utilização de imediato. Tal requer a utilização das tecnologias de comunicação disponíveis nos dispositivos móveis: Wi-Fi, Wi-Fi Direct e Bluetooth, e eventualmente a construção de redes que permitam *multihop* utilizando uma ou mais destas tecnologias.

Como foi mencionado no Capítulo 1, e será detalhado no Capítulo 3, os desafios de conectividade em redes de dispositivos móveis saem do âmbito desta dissertação e estão a ser endereçados por outras equipas de investigação no contexto do projeto onde esta dissertação se insere: *Hyrax: Crowd-Sourcing Mobile Devices to Develop Edge Clouds* [57].

2.2 Armazenamento de dados em redes de dispositivos móveis

O armazenamento de dados em redes de dispositivos móveis tem obrigatoriamente de respeitar algumas propriedades fundamentais para um bom funcionamento, uma vez que os dados não podem ser armazenados localmente em todos os dispositivos ou apenas num só dispositivo. Os sistemas de armazenamento distribuídos [18, 77] são desenvolvidos para permitir o acesso aos dados armazenados remotamente e mantendo as características essenciais como a disponibilidade, consistência e segurança dos dados, tolerância a falhas e o bom desempenho do sistema para o utilizador.

A disponibilidade dos dados é a característica que permite aos utilizadores obterem dados a qualquer momento, mesmo que tenha ocorrido uma falha no sistema. No nosso contexto pode acontecer quando um utilizador sai da rede por vontade própria ou por problemas de *hardware*. De forma a garantir uma boa disponibilidade, o sistema tem de ser tolerante a falhas. É possível haver diversos acontecimentos que levem um sistema de armazenamento distribuído a uma falha, como um problema de *hardware*, um *bug* no sistema, um corte de energia ou até uma catástrofe natural.

Segundo *Operating System Concepts* [77], uma das soluções mais utilizadas para garantir a toleraria a falhas e a disponibilidade dos dados é utilizando mecanismos de replicação. A replicação dos dados consiste em efetuar cópias e posterior manutenção dos dados em locais distintos do sistema, neste caso em dispositivos diferentes. Estando os dados replicados, mesmo que numa réplica o objeto em questão esteja inacessível ou corrompido, esse objeto estará disponível noutro(s) nó(s), assegurando as características referidas - tolerância a falhas e disponibilidade. O uso de replicação além de possibilitar que os dados no sistema não se percam, também permite aumentar a eficiência e o desempenho do sistema. Como os dados estão replicados por vários nós, é possível aceder de forma alternada aos nós que contêm os dados e além disso possibilita o acesso aos dados pela sua proximidade na rede, diminuindo a latência das operações. Isto pode permitir aos utilizadores realizarem as tarefas que desejam de forma eficaz e sem complicações causadas pelo sistema.

A consistência é a propriedade que garante que os dados possuem (ou irão convergir para) o mesmo estado em todas as réplicas, isto é, quando o sistema é distribuído e os

dados replicados, se existir modificação nos dados, esses dados devem ser modificados em todas as réplicas, não podendo ser possível aceder a dados inconscientes.

Relativamente à segurança dos dados, este é um assunto bastante discutido nos últimos anos. Grande parte dos utilizadores partilha dados importantes em sistemas de armazenamento deste tipo e permitir que esses dados sejam obtidos por terceiros é algo que se deve combater a todo o custo. Esta dimensão não será alvo de estudo por parte desta dissertação uma vez que já se encontra em estudo por outros trabalhos no âmbito do projeto *"Hyrax: Crowd-Sourcing mobile devices to develop edge clouds."* [57] no qual esta dissertação se insere.

2.3 Trabalho Relacionado

O armazenamento e partilha de dados em dispositivos móveis já foi estudado por outros autores e desenvolvidos alguns sistemas com o objetivo de permitir as tarefas de disseminação e armazenamento colmatando algumas das lacunas existentes.

Nesta secção apresentamos, na Subsecção 2.3.1, o estado da arte que se refere à investigação sobre disseminação e armazenamento de dados em redes de dispositivos móveis, onde teremos maior atenção em características como: o local e a unidade de armazenamento dos dados; persistência, disponibilidade e consistência dos dados; os mecanismos de resistência a falhas, se utilizarem; qual o mecanismo utilizado para a descoberta e obtenção de novos dados; a constituição da rede e, se utilizarem, o mecanismo de localização e encaminhamento dos dados. Inicialmente serão apresentados os sistemas e posteriormente será efetuada uma análise crítica sobre os mesmos.

Na Subsecção 2.3.3 é apresentado um levantamento das aplicações que estão disponíveis para descarregamento em lojas de aplicações como a *PlayStore* e a *AppleStore* relacionadas com o nosso sistema.

2.3.1 Trabalhos de investigação

PAN [51]. é um protocolo que permite fornecer armazenamento confiável em redes oportunistas onde vários utilizadores, representados por nós na rede, podem entrar e sair sem comprometer o desempenho do protocolo. Este protocolo usa um sistema de quóruns probabilísticos [54] e utiliza um protocolo *"gossip"* como forma de disseminação de mensagens. Esta solução permite garantir persistência, disponibilidade, e consistência dos dados e apresenta tolerância a falhas. Esta arquitetura permite que os utilizadores tenham acessos a escritas e leituras de forma concorrente.

Este sistema apresenta um grupo de nós especiais, denominado StS (Storage Set), que são selecionados previamente de forma aleatória, ou usando algoritmos específicos para essa funcionalidade com base na proximidade, por exemplo. A estes nós são atribuídas funções diferentes dos restantes pois são responsáveis pela tarefa de armazenamento e partilha dos dados na rede. Como se trata de uma rede oportunista formada por dispositivos

móveis, os elementos podem entrar e sair, obrigando o grupo de StS a ser reconfigurado.

Quando um cliente quer fazer uma atualização no sistema, comunica com um StS que irá transmitir essa informação para todos os restantes elementos do StS. Os dados não são considerados armazenados após a receção da primeira confirmação, é necessário receber o número de confirmações igual ao número do quórum, e só após essa receção o sistema tem garantia que esses dados foram armazenados e estão disponíveis para qualquer outro elemento do sistema. Logo, podemos concluir que os nós do StS são os locais onde os dados são armazenados e efetuam a replicação dos dados pelo quórum de StS, garantindo a persistência e a disponibilidade dos dados.

No caso das leituras, o quórum de leituras é menor do que o quórum de escritas uma vez que, normalmente, há mais leituras do que escritas. Quando o utilizador quer fazer uma leitura, envia o pedido para um elemento do StS mais próximo de si, denominado agente, e este comunica com os nós StS mais próximos para que estes lhe forneçam a sua versão desse objeto, respondendo assim ao cliente com a versão mais recente do objeto requisitado, garantindo consistência dos dados.

A utilização de um grupo especial que controla a leitura e escrita dos objetos obriga a que o sistema seja assimétrico, isto é, nós com responsabilidades fixas e diferentes. Neste caso todos os elementos do StS são responsáveis pelo armazenamento e partilha dos dados na rede, sendo que os dados devem estar armazenados em todos estes elementos.

iTrust [49]. tem como objetivo permitir a partilha, pesquisa, leitura e escrita de informações numa rede composta por dispositivos móveis (Android), que comunicam via Wi-Fi Direct. Quando um dispositivo pretende partilhar conteúdos envia mensagens para múltiplos dispositivos, utilizando inundação, que são escolhidos aleatoriamente, contendo essas mensagens metadados que possuem informação de onde está armazenado o conteúdo em causa. Esse conteúdo encontra-se armazenado no dispositivo de origem. Quando algum cliente pretende obter conteúdos, este envia pedidos para vários dispositivos remotos aleatoriamente. Quando um dispositivo recebe este pedido e contém os metadados pedidos, envia-os para o cliente em causa. Uma vez com os metadados, isto é, a localização do conteúdo em causa, o cliente faz o pedido diretamente à origem.

A rede deste sistema é descentralizada e nenhum nó apresenta papéis especiais quando comparado com outros.

A nível de características do sistema temos de ter em atenção o facto do iTrust não apresentar mecanismos de resistência a falhas, pois os objetos de cada nó apenas estão acessíveis no próprio nó e por isso, caso esse nó se ausente, saia do sistema ou tenha outro problema, os objetos que partilhou ficam inacessíveis. Esta mesma questão leva à inexistência de persistência dos dados na rede e influenciando a disponibilidade dos dados. Não existindo replicação dos objetos não existem problemas de consistência dos dados.

Phoenix [61]. é um sistema de armazenamento distribuído onde é pretendido garantir persistência e disponibilidade dos dados partilhados numa zona geograficamente bem definida utilizando redes de dispositivos móveis. É tomado em atenção o facto de estas redes serem um pouco inconstantes, isto é, os utilizadores podem sair e entrar da rede, e além disso estas redes não estão preparadas para a correção de falhas.

Este sistema funciona dividindo os dados a partilhar em blocos, dando uma maior facilidade de distribuição, e distribuindo-os por vários dispositivos, de forma a colaborarem entre si para armazenar esses blocos de forma mais eficiente.

Os utilizadores podem entrar no sistema com ou sem dados para partilhar. Quando um utilizador com dados para partilhar entra na rede, os seus dados são divididos em blocos e estes blocos são distribuídos pelos outros utilizadores da rede, mantendo k[1] cópias de cada bloco do objeto, mas cada nó apenas pode ter uma cópia de cada bloco. Estes blocos são transmitidos usando inundação, *flooding*, comunicando em *broadcast* com os restantes dispositivos.

O quórum é conseguido usando um mecanismo de rondas em que periodicamente e de forma assíncrona, cada nó seleciona um dos blocos que contêm e propagam essa informação para os restantes nós. Se não receber resposta de k nós com a informação que contém esse bloco, então o nó de origem transmite o bloco em causa para outros nós até que existam k ou mais nós com esse bloco. Caso contrário, dependendo da situação, ou ignora ou elimina o bloco pois existem k ou mais nós que contêm o bloco.

O facto de cada bloco estar replicado num quórum de elementos da rede fornece redundância dos dados o que permite persistência de dados e um mecanismo de resistência a falhas, que podem ser causadas por falhas da rede, tornando os dados sempre disponíveis no sistema.

Observando o comportamento do sistema, podemos considerar que a rede é não estruturada, descentralizada, e não existem nós com papéis especiais quando comparados com outros, pois todos possuem as mesmas funções.

Ao nível dos objetos partilhados, estes são imutáveis ou seja não existe a possibilidade de alteração dos mesmos, o que retira a preocupação com a consistência dos dados.

MobiTribe [85]. aborda uma solução para os problemas que procuramos resolver, utilizando uma rede não estruturada e onde todos os nós podem efetuar todas as tarefas disponíveis: produzir, armazenar ou obter dados. Sendo denominados neste sistema de *Criador*, *Ajudante* e *Consumidor*, respetivamente.

O *criador* é o dispositivo que pretende partilhar os dados. De forma a aumentar a disponibilidade dos dados na rede, esses dados são enviados para os *ajudantes*, utilizando um protocolo de encaminhamento pró-ativo, e comunicando através de *multicast*, ficando os dados armazenados no nó de origem e nos *ajudantes*. Os *ajudantes* encontram-se na

[1] K será o quórum do número de elementos da rede, isto é, k será igual ou maior a metade dos elementos da rede [53].

mesma rede do *criador* e, de acordo com padrões/estatísticas, devem tornar-se *consumidores* no futuro, logo vão necessitar dos mesmos. Ao replicar os dados pelos *ajudantes* o sistema obtém um mecanismo de resistência a falhas e também alcança persistência e maior disponibilidade dos dados para todos os elementos da rede.

Após o *criador* e os *ajudantes* possuírem os dados que o primeiro queria partilhar, os *consumidores* podem obter os dados do *criador* ou então aceder aos dados de um dos *ajudantes* utilizando o mecanismo de pergunta e resposta, diretamente a um destes nós, optando pelo nó mais próximo.

A decisão da escolha dos *ajudantes* segue um algoritmo ganancioso baseado em comunidades. Como referido, os nós podem estar em várias redes, chamadas comunidades, e essas comunidades encontram-se interligadas por *ajudantes*. Depois de identificadas as comunidades, é escolhido o nó que contém mais nós ligados a si, pois quantos mais nós estiverem ligados a si mais consumidores irá fornecer.

Comparativamente ao sistema que pretendemos desenvolver, esta solução apresenta um sistema de replicação semelhante a parte do nosso sistema de replicação, pois é utilizada uma previsão para descobrir quais os nós que irão consumir os dados e no nosso caso utilizamos os nós que obtiveram efetivamente os dados como réplica desse mesmo objeto, não estando a forçar os nós a armazenar ficheiros que não necessitam. A utilização do protocolo de encaminhamento pró-ativo é outra diferença comparativamente à nossa solução. Não pensamos em usar o protocolo pró-ativo pois este protocolo causa sobrecarga na rede, pois necessita do envio recorrente de mensagens na rede para identificação dos nós ativos, e um dos nossos objetivos é evitar sobrecarga da rede. Além disso, este protocolo não apresenta facilidade em lidar com o crescimento do número de nós na rede, pois com o aumento da rede, maior será a sobrecarga sobre a mesma.

Krowd [21]. foi desenvolvido no contexto deste projecto "*Hyrax: Crowd-Sourcing mobile devices to develop edge cloud.*" [57]. Trata-se de um sistema de armazenamento baseado em chave-valor direcionado principalmente para eventos ao vivo onde os utilizadores pretendem partilhar fotos e vídeos. Este sistema tem como objetivo permitir que utilizadores que se encontrem próximos, numa vizinhança, possam partilhar e obter dados dos seus vizinhos, utilizando *tags* como identificador dos dados.

A rede que suporta este sistema é construída utilizando um ponto de acesso que utilizando Wi-Fi ou Wi-Fi Direct constrói e monitoriza a rede, funcionando o sistema apenas na rede respetiva.

A novidade deste trabalho é fundamentalmente a proposta de um novo tipo de tabela de dispersão (TD) que permite facilitar a localização e obtenção dos dados neste tipo de redes.

Quando um utilizador quer partilhar um ficheiro, envia uma mensagem utilizando *broadcast* para todos os nós da rede, para que estes tomem conhecimento, adicionando na sua tabela de dispersão, o par com a *tag*, que identifica o objeto, e o nó de origem.

As tabelas de dispersão, para manterem uma lista dos nós ativos na rede, todos os nós a cada 5 segundos, enviam *beacons UDP* usando *broadcast* permitindo assim que todos os dispositivos tenham conhecimento da sua presença. Caso o dispositivo tenha deixado de enviar estes *beacons*, ou quaisquer outros dados, é forçado um *ping* para verificar se o dispositivo ainda é contactável. Caso esta ação falhe o dispositivo é considerado fora da vizinhança, isto é, pode ter saído da rede ou ficado sem bateria por exemplo.

Quando um utilizador necessita de obter um objeto, executa a função de dispersão, na sua tabela de dispersão, com a *tag* do objeto em causa, de forma a obter qual o nó que tem de comunicar de forma a obter o objeto em causa[2]. Caso o nó seguinte não contenha os dados, esse nó indicará qual o nó que se deve seguir para a obtenção do objeto.

A utilização deste sistema apenas é recomendada para uma única rede, não sendo desenvolvido para suportar múltiplas redes. Podemos comparar com o que pretendemos obter dizendo que este sistema apenas pode ser utilizado em cada célula do nosso sistema e não num conjunto de células.

Existem algumas considerações que são necessárias fazer sobre este sistema. Uma das questões é o facto de a disponibilidade dos dados poder ser afetada pois os objetos apenas estão armazenados no nó de origem, não existindo mecanismos de replicação, o que leva a outro problema: a não tolerância a falhas quer da rede quer dos dispositivos. Não existindo replicação, a persistência dos dados fica limitada, mas evita a preocupação com a consistência dos dados.

Ephesus [79]. é um sistema de partilha e armazenamento de dados previamente desenvolvido no contexto deste projeto *"Hyrax: Crowd-Sourcing mobile devices to develop edge cloud."* [57], sendo de certa forma, uma versão mais restrita do trabalho que se pretende desenvolver nesta dissertação.

O Ephesus trata-se de um sistema de armazenamento descentralizado e distribuído, do estilo *key-value*. É direcionado para redes de dispositivos móveis ligados por Wi-Fi usando um ponto de acesso, infraestrutura ou *hotspot*, garantindo escalabilidade, onde é possível compartilhar dados entre dispositivos que estejam geograficamente próximos, de forma assíncrona. Este sistema é efémero pois só existe enquanto há dispositivos interligados que o suportem. Logo, quando o último dispositivo se desconectar, o sistema deixará de existir.

Este sistema não necessita de qualquer ligação à Internet para efetuar as suas funcionalidades, apesar disso necessita de um ponto de acesso ou de um dispositivo móvel que faça de *hotspot* para que o sistema funcione. A rede trata-se de uma rede não estruturada e simétrica, isto é, não existem nós com papéis especiais comparativamente com outros.

A obtenção de novos objetos é alcançada utilizando o modelo pedido-resposta, onde é utilizada a tabela de dispersão distribuída (TDD) para fornecer qual o melhor caminho para chegar ao nó responsável por esse objeto.

[2] Esta solução foi baseada em *"highest random weight hashing"* [83].

O mecanismo de replicação utilizado por este sistema é baseado em popularidade, utilizando replicação ativa e passiva. Para cada objeto partilhado no sistema, são escolhidos um grupo de nós para gerir a replicação desse objeto. Aquando da entrada no sistema de um novo objeto, é escolhido um número predefinido e de forma aleatória os nós que devem conter réplicas desse objeto, representando a replicação ativa. Ao longo do funcionamento, os dados vão ser obtidos por outros elementos da rede, ficando mais populares, e é neste momento que os nós responsáveis ao perceber que existem mais cópias do objeto no sistema, utilizam a replicação passiva, para reduzir a replicação ativa, mantendo sempre o número predefinido de cópias do objeto. Isto permite que o sistema adquira resistência a falhas e consiga armazenar os dados de forma persistente dentro da rede, tornando-os disponíveis a qualquer momento para serem obtidos por outros nós.

Como caso de estudo foi desenvolvida uma aplicação Android, "Shared photo gallery", que permite a partilha de fotos utilizando o sistema Ephesus. Os resultados experimentais efetuados pelos autores comprovaram a usabilidade do sistema assim com o seu baixo consumo energético.

Este trabalho pode ser considerado o trabalho mais semelhante ao pretendido por esta dissertação pois os nossos objetivos são semelhantes aos apresentados pelo Ephesus mas no nosso caso não utilizaremos o modelo de pedido-resposta para obtenção de dados pois queremos desacoplar a tarefa de emissor e recetor de dados, aplicando o mecanismo de interação publicador/subscritor. Outro ponto que diferencia a nossa solução é a utilização de uma tabela de dispersão geográfica baseada em células o que permite melhor localização dos nós e dos dados comparativamente à TDD, permitindo ainda melhor resistência à entrada e saída de nós na rede. Sendo uma *key-value* é possível obter os dados que foram publicados no sistema (e não foram perdidos), independentemente de quando o foram, algo semelhante ao desejado para o nosso sistema.

2.3.2 Discussão crítica

Nesta subsecção iremos apresentar uma análise crítica às soluções apresentadas anteriormente. Para uma melhor compreensão, esta análise crítica foi dividida em quatro pontos principais: **disponibilidade e tolerância a falhas**, **unidade de replicação**, **mecanismos de obtenção de dados**, **consistência dos dados**, **encaminhamento** e **tarefas dos nós**. Podemos verificar algumas diferenças na Tabela 2.1.

Disponibilidade e tolerância a falhas. Todos os sistemas estudados que endereçam esta questão fazem-no através de mecanismos de replicação. A abordagem mais comum é replicação ativa, em que o próprio sistema envia cópias dos dados para outros nós, sem consentimento dos utilizadores. O nosso sistema também pretende fazer uso de tal mecanismo, mas, à semelhança do Ephesus [56, 79], também queremos aproveitar as réplicas existentes no sistema que surgem pelos descarregamentos dos utilizadores, de forma a aproveitar ao máximo as cópias dos dados existentes no sistema, sendo esta abordagem

Tabela 2.1: Resumo dos sistemas existentes.

	PAN	**iTrust**	**Phoenix**	**MobiTribe***	**Krowd**	**Ephesus**	**Thyme**
Encaminhamento	Inundação	Inundação	Inundação	Encaminhamento pró-ativo	Encaminhamento pró-ativo usando TD	Encaminhamento pró-ativo usando TDD	Encaminhamento geográfico
Unidade de armazenamento	Objeto	Objeto	Blocos	Objeto	Objeto	Objeto	Objeto
Replicação	Sim	Não	Sim	Sim	Não	Sim	Sim
Disponibilidade dos dados	Sim	Não	Sim	Sim	Não	Sim	Sim
Consistência dos dados	Sim	Sim	Sim	Sim	Sim	Sim	Sim
Obtenção dos objetos	Pedido-Resposta	Pedido-Resposta	Pedido-Resposta	Pedido-Resposta	Pedido-Resposta	Pedido-Resposta	Publicador/ Subscritor
Resistência a *churn*	Replicação	Não	Replicação	Replicação	Não	Replicação	Replicação
Nós com papéis especiais	Sim	Não	Não	Não	Não	Não	Não

conhecida como replicação passiva. Em relação à replicação ativa, com exceção do MobiTribe [84, 85], nenhum dos restantes utiliza a questão da proximidade para replicar os dados. Na nossa solução pretendemos utilizar a proximidade para efetuar uma replicação local, de forma a evitar a transferência de grandes quantidades de dados pela rede, algo que acontece nos sistemas onde a replicação não é controlada.

Unidade de replicação. A unidade de replicação mais comum nos sistemas que endereçam replicação é o objeto, contudo existem sistemas como o Phoenix [61] que optaram pela divisão dos objetos em blocos e utilizar esses blocos como unidade de replicação. A nossa abordagem inicial será a utilização do objeto como unidade de replicação, em que haverá um balanceamento dos pedidos pelas várias réplicas. Pensamos ser a solução mais adequada, uma vez que pretendemos que este sistema seja utilizado como suporte para armazenamento e partilha de ficheiros, facilitando a descoberta e obtenção dos mesmos. Eventualmente, poderão estudar-se esquemas de replicação parcial quando o número de dispositivos for maior do que o número de ficheiros a replicar.

Mecanismos de obtenção de dados. O mecanismo para obtenção de dados usado em todas as soluções anteriores é o pedido-resposta, que consiste em enviar pedido(s) quando necessita de informação, para um ou mais nós e esperar pela resposta do(s) nó(s). No nosso caso achamos que a melhor solução será a utilização do mecanismo publicador/subscritor, pois este mecanismo fornece-nos desacoplamento entre os nós que produzem os dados e os nós que os consomem. Também permite garantir que os utilizadores do nosso sistema obtenham todos os dados relacionados com os eventos que eles estão interessados, sendo que com a utilização deste mecanismo apenas é necessário efetuar a subscrição e depois desse momento receberá todo o conteúdo relacionado.

Consistência dos dados. A existência de replicação poderia levar a problemas de inconsistência mas, tal como no Phoenix [61], decidimos que os nossos objetos seriam imutáveis, logo os objetos replicados não serão alterados, eliminando assim os problemas de inconsistência que poderiam vir a surgir. Outra solução seria a utilização de consenso de versões

entre nós, como no PAN [51], mas esta solução obrigaria a carregar a rede com mais troca de mensagens. Em relação aos metadados dos objetos, estes podem ser alterados. Quando os metadados são alterados, essa informação será disseminada pelos restantes elementos que contêm réplicas dos metadados, mantendo a consistência.

Tarefas dos nós. Excluindo o PAN [51], em todos os outros sistemas não existem nós com papéis especiais, pois todos os nós podem efetuar qualquer operação dentro do sistema. Esta é uma abordagem que queremos adotar para o nosso sistema pois achamos que possuir nós exclusivamente responsáveis pelo armazenamento dos dados, como acontece no PAN [51], é discriminatório podendo causar problemas de sobrecarga aos nós, quer a nível ou tarefas.

Encaminhamento. Dentro dos sistemas que utilizam encaminhamento, o protocolo mais comum é o pró-ativo, em que os nós contêm tabelas onde estão armazenados os caminhos conhecidos dentro da rede. No nosso caso pretendemos utilizar um protocolo de encaminhamento geográfico, aproveitando a capacidade de localização dos dispositivos móveis, de forma a diminuir a sobrecarga sobre a rede, a suportar a entrada e saída de elementos da rede, permitir escalabilidade e a apresentar encaminhamento dos pacotes de forma eficiente.

2.3.3 Aplicações disponíveis comercialmente

Existem aplicações no mercado, em lojas de aplicações, que permitem a troca de informação direta entre dois dispositivos ou um grupo de dispositivos, tentando assim evitar que sejam usadas *cloud* remotas para efetuar esta troca de dados.

As aplicações *SuperBeam* [48], *Xender* [3], *Qikshare* [41] e *Drop* [23], entre outras disponíveis para Android, permitem exatamente essa funcionalidade. Com estas aplicações, o utilizador do smartphone de origem seleciona os dados que pretende enviar e coloca-se na operação de "envio" e espera que outros smartphones que estejam interessados se coloquem na sua rede. Os smartphones interessados colocam-se na operação "receção" e se se encontrarem na área de envio do dispositivo de origem, recebem os dados que a origem enviou, de forma síncrona. No caso do *Xender* [3] a rede é limitada pelo alcance do seu Wi-Fi, uma vez que ele cria um ponto de acesso móvel, já o *SuperBeam* [48] e o *Drop* [23] podem funcionar também criando um ponto de acesso móvel ou por Wi-Fi Direct.

Estas aplicações permitem transferir dados entre um grupo de dispositivos, mas os dados que são partilhados são escolhidos pela origem e os recetores não têm qualquer opção de escolha, ou decidem receber os dados ou não, não podem eles próprios escolher os dados que querem receber.

A aplicação *FrostWire* [32], disponível para dispositivos Android, utiliza o mecanismo de comunicação Wi-Fi. O seu funcionamento é simples, o nó de origem seleciona um

número de ficheiros, à sua escolha, que pretende partilhar e possibilita que outros dispositivos vejam e façam *download* de parte ou da totalidade dos ficheiros que partilhou. Esta aplicação resolve a questão da escolha de dados por parte dos dispositivos que pretendem receber a informação, mas isto é apenas possível entre dois dispositivos, não existindo uma pasta partilhada entre todos os elementos do grupo.

MediaBowl: FileShare [67] permite que o nó de origem partilhe dados e os restantes dispositivos presentes na rede decidem quais os dados que querem aceder, mas apenas é possível a opção de leitura sobre esses dados. Esta aplicação apenas permite a partilha de dados de um dispositivo de origem, sendo de comunicação 1-M. Além disso é importante notar que este sistema é só de visualização, ou seja, é possível visualizar o objeto, mas não é possível fazer *download* dos mesmos. Estes são pontos divergentes em relação ao pretendido nesta dissertação.

A Tabela 2.2 apresenta um resumo das aplicações descritas anteriormente em relação às características básicas que pretendemos para este projeto, sendo que a última linha apresenta quais as características que pretendemos para o nosso sistema.

Tabela 2.2: Comparação das aplicações disponíveis comercialmente.

Aplicação	Partilha assíncrona	Comunicação	Necessidade de infraestrutura	Armazenamento
SuperBeam	Não	1-M	Não	Não
Xender	Não	1-M	Não	Não
FrostWire	Sim	1-1	Não	Temporário[1]
MediaBowl	Sim	1-M	Não	Temporário[1]
Qikshare	Não	1-1	Não	Não
Drop	Não	1-1	Não	Não
THYME	Sim	N-M	Não	Sim

O nosso objetivo é desenvolver uma aplicação que permita a troca de dados de forma assíncrona, de forma a permitir armazenamento dos dados e que esses dados estejam disponíveis durante o tempo de vida da rede. Existe também a necessidade de ser uma comunicação de muitos para muitos (N-M) de forma a que vários utilizadores possam escrever e ler a qualquer momento, sendo que também é importante que esta aplicação possa funcionar sem infraestrutura de suporte.

Analisando a tabela anterior podemos chegar à conclusão que nenhuma das aplicações estudadas cumpre o conjunto de características que pretendemos.

2.4 O Modelo Publicador/Subscritor

O modelo publicador/subscritor trata-se de um modelo de interação reativa, onde um evento no sistema, desencadeia uma ação que na maioria dos casos é uma comunicação para o cliente [18, 27].

[1] Existe armazenamento enquanto o dispositivo de origem está conectado. Mas quando este se desconecta os dados são perdidos.

Este paradigma permite que os utilizadores sejam subscritores ou produtores, podendo representar diferentes papéis ao longo do tempo. Quando um cliente está interessado em receber dados relacionados com um ou vários eventos, subscreve esse ou o conjunto de eventos, *"subscribe()"*. Quando os produtores publicarem informações sobre esse(s) eventos no sistema, *"publish()"*, o sistema encarrega-se de notificar os utilizadores que manifestaram interesse nesse(s) evento(s), *"notify()"*, isto é, subscreveram o evento em questão. Podemos ver um sistema básico de publicador/subscritor na Figura 2.1.

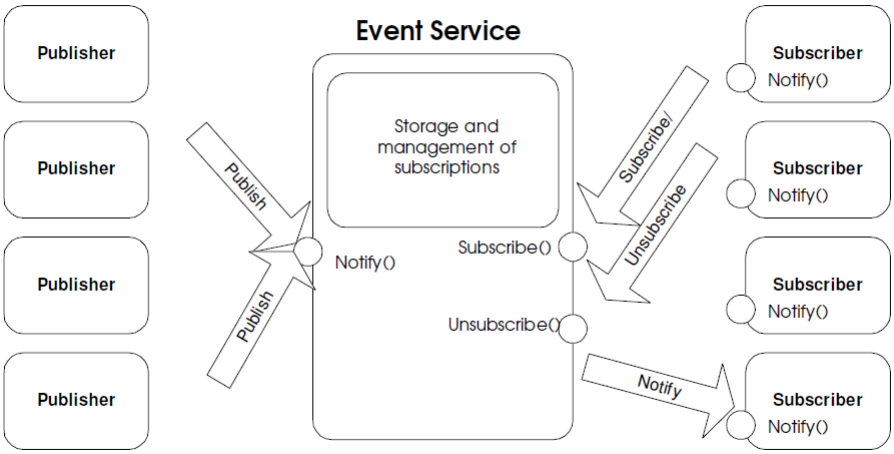


Figura 2.1: Sistema básico de publicador/subscritor [27].

Este modelo de interação tem algumas características interessantes que o tornam bastante útil. Os consumidores e os produtores podem variar ao longo do ciclo de vida do sistema e este é suficientemente eficiente para se organizar e otimizar. Os consumidores não têm qualquer controlo sobre os dados que são produzidos, pois os produtores produzem os dados independentemente dos consumidores e os consumidores consomem todos os eventos que são produzidos independentemente dos produtores, desde que tenham subscrito os eventos anteriormente, desacoplando assim as duas tarefas.

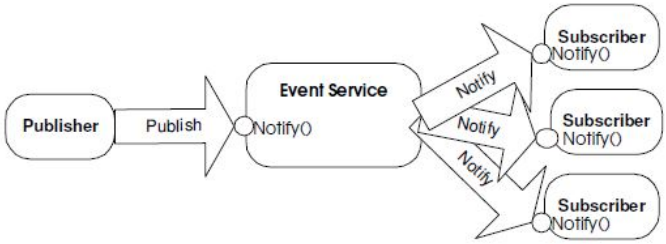
O modelo publicador/subscritor padrão permite desacoplar a ligação entre publicadores e subscritores que são bastante interessantes para serem aplicadas em redes de dispositivos móveis, como o desacoplamento espacial, desacoplamento temporal e desacoplamento da sincronização.

Desacoplamento espacial A comunicação entre publicadores e subscritores é anónima, isto é, os publicadores publicam os dados sem saber quem são os subscritores do evento em questão e os subscritores recebem os dados relativos aos eventos subscritos sem terem referência de quem foi o publicador dos dados que receberam.

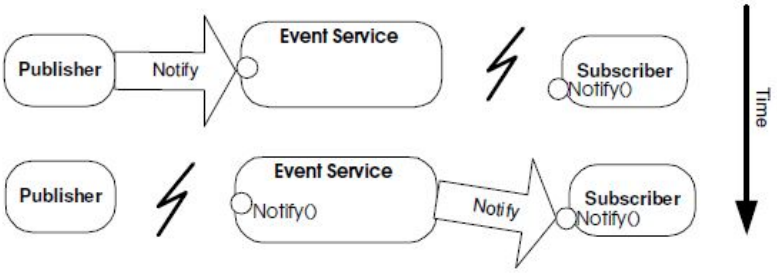
Desacoplamento temporal Não existe a necessidade de as duas partes, publicador e subscritor, estarem ativas ao mesmo tempo para as operações terem sucesso. O publicador pode publicar no sistema mesmo que não existam subscritores, o subscritor

pode subscrever mesmo que não existam publicadores e o subscritor pode ainda receber notificações mesmo que o publicador esteja desconectado, independentemente do tempo.

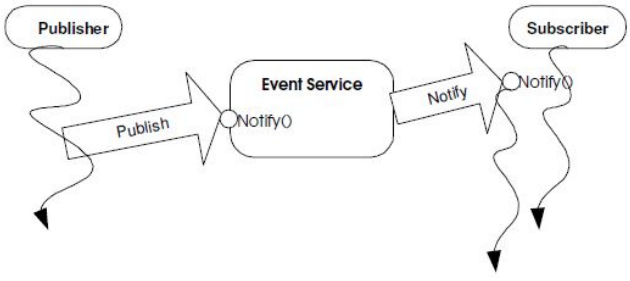
Desacoplamento da sincronização A comunicação é assíncrona, isto permite que os dispositivos possam efetuar publicações e subscrições sem serem bloqueados de receber notificações. Isto é possível porque a produção de dados e receção de notificações são tratadas em fluxos diferentes, permitindo uma comunicação assíncrona.



a Desacoplamento espacial.



b Desacoplamento Temporal.



c Desacoplamento da sincronização.

Figura 2.2: As três dimensões do desacoplamento do publicador/subscritor [27].

Estas características de desacoplamento entre as duas funções permitem aumentar a escalabilidade no contexto desta dissertação, pois como não existem dependências entre os produtores e consumidores, não existem problemas com a constante entrada e saída do sistema, muito característico neste contexto, o que poderia ser muito complicado noutros tipos de modelos de interação.

2.4.1 Publicador/subscritor em ambientes móveis

Ao longo dos anos, foram vários os investigadores que desenvolveram sistemas usando o modelo publicador/subscritor para permitir ou melhorar a disseminação de informações em redes oportunistas utilizando dispositivos móveis.

O trabalho descrito em [39] aborda um sistema onde é utilizado o modelo publicador/subscritor modificado de forma a ser utilizado em ambientes móveis. O modelo desenvolvido propõe que os elementos da rede sejam divididos em 3 funções distintas: publicar objetos em eventos, armazenar as subscrições e partilhar os objetos, ou subscrever os eventos. A utilização deste modelo possibilita que a comunicação da publicação dos dados seja efetuada usando *broadcast* ou *multicast*.

Utilizando *brodcast*, o publicador envia uma mensagem para todos os elementos que contêm as subscrições, informando que publicou um objeto. Caso algum desses nós possua subscrições relativas ao evento(s) desse objeto, será enviado o objeto para os nós subscritores.

Se a rede for complexa, será um pouco pesado enviar os eventos para todos os nós que contêm as subscrições. Seguindo esta ideia foi utilizada a comunicação *multicast*. Neste caso, apenas os nós que contêm a subscrições relativas ao(s) evento(s) vão receber a informação da publicação do novo objeto, reduzindo assim a carga na rede e facilitando as comunicações.

As redes oportunistas apresentam uma topologia bastante irregular, com entrada e saída de elementos da rede, o que pode levar a que a comunicação entre o subscritor e o nó que armazena a sua subscrição seja perdida. Quando a conexão for restabelecida o nó que armazena a subscrição pode ser o mesmo ou não, sendo outro nó. No caso de ser outro, é necessário que o nó antigo transmita ao novo nó quais as suas subscrições e quais os pacotes que se encontram por enviar ou, outra opção, será o próprio subscritor conter as suas subscrições e o último pacote que recebeu, evitando assim a comunicação entre os dois nós que armazenam subscrições. Em ambos os casos é necessário reenviar os objetos que não foram recebidos durante a desconexão. Porém este número de mensagens pode ser muito elevado, causando grande largura de banda e possivelmente problemas de armazenamento aos nós que armazenam subscrições, por isso é proposta uma solução no qual as assinaturas têm uma validade que permite ao nó de armazenamento apagar esses dados caso seja ultrapassada essa validade.

Os autores tentaram otimizar o método de partilha de eventos para redes móveis resolvendo alguns problemas críticos neste tipo de redes, como a desconexão e a entrada e saída de elementos da rede, desenvolvendo soluções distribuídas, mas também soluções centralizadas que não foram discutidas aqui pois não se enquadram no nosso objetivo.

O funcionamento do modelo publicador/subscritor descrito em [39] é comportamento comum nos sistemas atuais, como por exemplo em [62] e [19] onde são apresentados sistemas semelhantes de forma a resolver problemas relacionados com a mobilidade dos nós,

a interação entre dispositivos, o processamento descentralizado e a possibilidade de existência de um número elevado de nós. Resumindo, estes artigos apresentam soluções para a adaptação do modelo de interação publicador/subscritor em sistemas móveis usando redes oportunistas.

Com o crescimento da utilização deste modelo de interação em redes oportunistas foram surgindo novas necessidades e desafios que levaram a novas investigações para tentar colmatar essas lacunas. Exemplos representativos de trabalhos que procuraram resolver algumas lacunas existentes são: [13] - tenta minimizar os problemas causados pela mobilidade dos nós e mudança de topologia da rede, [30] - tenta resolver problemas causados pelo tipo de ambiente *ad hoc*, [7, 17] - procuram encontrar metodologias de descoberta e envio de dados em redes *ad hoc*, [31] - debruça-se sobre a disseminação dos dados sem causar sobrecarga nos nós em redes oportunistas, e [25] - apresentam como aplicar e configurar um sistema de publicador/subscritor direcionado para IoT.

2.4.2 Obtenção de dados no passado

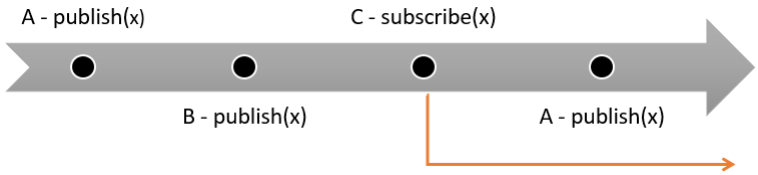
O modelo de interação publicador/subscritor padrão apresenta características que pretendemos para o nosso sistema porém também apresenta características que condicionam funcionalidades importantes do nosso sistema. O modelo padrão apenas permite que os subscritores recebam os dados que serão produzidos após a subscrição, mas o nosso objetivo será permitir que os subscritores obtenham todos os dados relativos a esses eventos independentemente do momento da subscrição.

Supondo que os utilizadores A e B produzem eventos sobre o evento "X", no momento 0, 1 e 3. Segundo o modelo publicador/subscritor padrão, se o utilizador "C" subscrever o evento "X" no momento 2 apenas irá receber os dados publicados no momento 3 (Figura 2.3a). Mas no nosso caso, queremos que "C" receba todos eventos publicados com o evento "X", dentro do intervalo de tempo que o utilizador definiu quando efetuou a sua subscrição. No caso do exemplo da Figura 2.3 seria desde o início do evento até ao final do mesmo (Figura 2.3b).

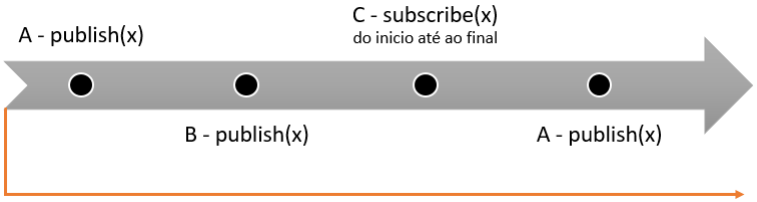
O sistema que esta dissertação aborda necessita que o mecanismo de publicador/subscritor possibilite a obtenção de dados que já aconteceram, isto é, é necessário permitir que ao efetuar uma subscrição seja possível obter os dados que já foram produzidos relativos a esse evento. Em seguida apresentamos alguns trabalhos que partilham este mesmo objetivo.

A solução encontrada em [15, 28, 47] para garantir esta característica consiste em armazenar, temporariamente ou não, os dados em algum dos nós da rede para possibilitar que estes sejam resgatados a quando da subscrição.

A solução proposta por [15] para permitir a obtenção de dados no passado consiste em criar *bufferes* que armazenam os dados que foram encaminhados por esses nós. A existência destes *bufferes* obriga a que os pedidos de subscrição sejam alterados, uma vez que neste momento é necessário saber a *tag* do evento, mas também é necessário



a . Subscrição padrão



b . Subscrição no passado

Figura 2.3: Comparação entre subscrição padrão e pretendida.

saber quantos objetos passados relativos a esse evento o utilizador pretende receber. Logo, quando o cliente pretende fazer uma subscrição terá de enviar a *tag* e o número, n, que será o número de objetos, produzidos no passado, que pretende obter. Assim quando um novo cliente faz uma subscrição, em primeiro lugar recebe os n objetos produzidos passados e só depois receberá os eventos em tempo real.

Nesta solução, os objetos estão espalhados pelos vários nós do sistema e cada nó pode conter objetos de vários eventos, por isso, cada nó contém um *buffer* para cada evento, *tag*, que conhece, evitando assim que exista armazenamento de informação de forma aleatória.

O facto de os objetos poderem estar em vários nós pode levar a que existam dados replicados a quando da subscrição do novo nó. Para resolver este problema são utilizados nós especiais que recebem todos os objetos com a respetiva *tag*, ordenando-os e eliminando os duplicados. Quando os objetos estiverem tratados serão enviados para o nó que efetuou a subscrição.

A solução proposta em [28, 47] é bastante semelhante à anterior. Trata-se de um sistema distribuído mas que utiliza uma rede com dispositivos fixos. Neste caso os objetos, quando publicados, são armazenados em bases de dados históricas que guardam todos os dados que são publicados. Ao contrário da solução anterior, os dados relativos a uma *tag* apenas estão guardados numa base de dados, onde várias bases de dados podem armazenar várias *tags*, de forma a maximizar o armazenamento.

Com esta solução, quando o cliente efetua a subscrição e se esta trouxer a informação que é necessária para obter os dados do passado, que neste caso é representado por um intervalo de tempo, esta assinatura será dividida, sendo a assinatura normal tratada da forma padrão e a assinatura temporal originará um envio de um pedido em direção à base de dados histórica que armazena os dados da assinatura pretendida. O pedido chegará à base de dados respetiva pois, cada nó contém uma tabela SRT (*Subscription Routing Table*) que contém o nó seguinte para onde deve ser encaminhado o pedido para que chegue ao

seu destino. As bases de dados ao receberem a assinatura recuperam os dados necessários e enviam diretamente para o assinante.

Analisando, a obtenção de dados no passado usando o modelo publicador/subscritor, é efetuada armazenando os objetos em locais dentro do sistema, nos próprios nós como em [15] ou em bases de dados segundo [28, 47], para que estes sejam obtidos pelos novos subscritores a quando da sua subscrição. Além disso, o pedido da subscrição tem de ser alterado pois é necessário informar o sistema que o subscritor pretende dados do passado. Essa informação pode ser transmitida usando diretamente o número de objetos que quer receber, como em [15], que pode ser complicado se o utilizador não souber o número de objetos que já foram publicados, ou usando um intervalo de tempo, como em [28, 47], o que nos parece mais correto comparativamente à solução anterior.

Concluindo, no nosso sistema iremos utilizar os próprios nós para armazenar todos os dados que são publicados no sistema, para que seja possível os novos subscritores obterem esses dados. Iremos adicionar ao pedido de subscrição um intervalo de tempo que definirá quais os objetos já publicados, que se pretende obter.

2.5 Descoberta e localização dos dados em redes de dispositivos móveis

Quando estamos perante um sistema de armazenamento distribuído, os dados estão distribuídos pelos vários nós, sendo necessário saber a sua localização e qual o caminho a percorrer na rede de forma a alcançar o nó que pretendemos. No caso de se estar perante uma rede de dispositivos móveis, é possível utilizar todos os nós como intermediário, mas é necessário lidar com características como a mobilidade dos nós, os problemas de interferência na rede, limitações de energia ou a existência de rotas redundantes, o que pode levar à necessidade de adaptação de caminhos podendo ser difícil a localização dos dados e a descoberta do caminho a percorrer pelos pacotes até atingir o nó final, sendo necessário recorrer a algoritmos/protocolos de encaminhamento (*routing*) específicos.

2.5.1 Encaminhamento

O encaminhamento [9, 44], isto é, como é adquirido o caminho que cada pacote deve seguir até ao seu destino, pode ser bastante diferente dependendo do contexto em questão e das características que pretendemos aplicar ao sistema. Os protocolos de encaminhamento em redes de dispositivos móveis podem ser divididos em cinco grandes grupos: geográficos, pró-ativo, reativo, híbridos e hierárquicos.

Os algoritmos de encaminhamento geográficos [55] são usados quando é possível ter conhecimento da localização dos nós da rede, quando o nó de origem tem conhecimento da sua própria localização assim como a localização dos seus vizinhos próximos e do nó de destino. Se o nó de origem conhecer a sua posição e a posição do nó de destino, então pode enviar o pacote em direção a esse destino, utilizando os seus vizinhos mais próximos

do nó de destino. Isto evita que seja necessário armazenar os caminhos já existentes, assim como fazer pesquisas em toda a rede de forma a descobrir o caminho desde o nó de origem até ao nó de destino.

No protocolo pró-ativo, também conhecido como *Table-Driven*, os nós contêm tabelas onde estão armazenados os caminhos conhecidos dentro da rede. Eles têm o conhecimento da topologia da rede, e consequentemente os caminhos existentes, através de trocas constantes de informação sobre a topologia da rede entre os nós.

O protocolo reativo, ou *On Demand*, os nós não armazenam os caminhos da rede, isto é, sempre que for necessário enviar um pacote, o nó determina no momento do envio, qual o caminho que esse pacote deve seguir. Para descobrir qual o caminho que o pacote deve seguir, o nó que contém o pacote envia um pedido para toda a rede, efetuando um pedido *broadcast*, uma inundação na rede, esperando que os restantes nós lhe respondam de forma a criar um caminho possível.

O protocolo de encaminhamento híbrido consegue obter características dos dois algoritmos anteriores, pró-ativo e reativo. Muitas das vezes estamos perante uma necessidade para qual um dos algoritmos não satisfazem totalmente as necessidades. Neste caso o melhor é usar um protocolo híbrido, pois são protocolos que incluem características, boas ou más, de ambos os protocolos anteriores e podem adaptar-se melhor às características de uma determinada rede.

O protocolo de encaminhamento hierárquico consiste na agregação dos diversos nós em grupos (*clusters*) e dentro de cada *cluster* existe um nó especial, denominado *clusterhead*. Estes nós são responsáveis por encaminhar os pacotes para o *cluster* onde se encontra o nó de destino, comunicando com o *clusterhead* do *cluster* em questão. Apenas estes nós têm conhecimento dos restantes grupos e podem efetuar essa comunicação.

Tabela 2.3: Comparação dos algoritmos de encaminhamento.

	Geográficos	**Pró-ativo**	**Reativo**	**Hierárquicos**
Localização dos nós	Sim	Não	Não	Não
Latência	Baixa	Baixa	Alta	Baixa
Sobrecarga da rede	Baixa	Alta	Baixa	Baixa
Funcionamento em redes complexas	Sim	Não	Não	Sim mas estruturada de forma hierárquica
Suporte *churn*	Sim	Não	Sim	Não

O protocolo de encaminhamento geográficos é, dos referidos, o que mais nos agrada para o desenvolvimento deste sistema. Este protocolo apresenta a vantagem de saber a localização dos nós no sistema, facilitando o envio dos dados, algo que não é possível conhecer com a utilização de qualquer um dos outros. A nível de latência os valores são baixos pois não é necessária a pesquisa do caminho como no protocolo reativo. A sobrecarga da rede também é uma vantagem deste protocolo pois apenas as mensagens do sistema são enviadas pela rede, contrariamente ao protocolo pró-ativo onde é necessária

disseminação constante de informação para que seja possível manter as tabelas atualizadas. O protocolo geográfico funciona de forma eficiente em redes complexas, com bastantes nós e onde a rede não esteja bem estruturada. O facto deste tipo de protocolos usar informações geográficas possibilita ainda que os sistemas respondam com mais facilidade às modificações das rotas e se tornem mais escaláveis quando comparamos com os restantes protocolos.

Em contrapartida, este protocolo apresenta a desvantagem de necessitar da obtenção da localização geográfica dos nós. Mas, atualmente, utilizando dispositivos móveis, como smartphones e tablets, é possivel obter essa informação. Em grande parte desses dispositivos é utilizado o GPS [1] de forma a obter a sua localização geográfica, mas também pode ser obtida a sua posição usando pontos de referência de algum sistema de coordenadas fixo.

2.5.2 Localização dos dados

Como referido anteriormente, no contexto em que nos encontramos, os dados estão distribuídos pelos diversos nós da rede, logo quando necessitamos de enviar um pacote - um pedido para obtenção de dados por exemplo - para um determinado nó é necessário saber qual a sua localização na rede. Neste caso específico, usando um protocolo geográfico é necessário saber a localização geográfica do nó.

Existem soluções que permitem saber a posição do nó que pretendemos, como por exemplo, em [21] é utilizada uma tabela de dispersão (*hash table*), que indica qual o nó que se deve seguir para chegar ao nó de destino. Esta solução é interessante, mas apenas para redes de pequenas dimensões, pois não sendo distribuída, é difícil manter a informação de todos os nós na tabela. O facto desta tabela de dispersão não ser distribuída também traz o problema de não ser possível resistir à mudança de topologia da rede. O facto de ser necessário que todos os nós comuniquem com os restantes, repetidamente com um intervalo de tempo relativamente curto, de forma a indicar que estes ainda estão dentro do sistema, também pode ser um problema. O envio repetidamente destas mensagens, em redes com maior volume, causa uma sobrecarga bastante grande à rede e além disso o tráfego de dados na rede iria ser bastante elevado.

Já no artigo [79], é utilizada uma tabela de dispersão distribuída (TDD) para a localização dos nós, onde os nós têm conhecimento de parte da rede, isto é, a localização de alguns dados. Esta solução apresenta resultados bastante interessantes a nível de escalabilidade de dispositivos. Porém, a nível de modificação da topologia da rede, com entrada e saída de dispositivos, esta solução apresentou resultados não muito satisfatórios, pois não sabendo a localização exata dos nós caso estes se movimentem ou algum nó do "caminho" se desligar, obrigaria a procurar um novo caminho. Além disso é necessário manter uma rede estruturada o que pode apresentar custo para o sistema quando falamos de um

[1] O GPS é um sistema de localização que utiliza 32 satélites distribuídos em 6 órbitas distintas de forma a obter a localização física atual, usando triangulação [44].

grande número de nós, o que poderia ser melhorado utilizando um TDG com noção de localização.

Utilizando a localização geográfica dos nós, através dos mecanismos de localização como o GPS, é possível localizar todos os nós dentro da rede. Com essa informação, conseguimos construir uma tabela de dispersão geográfica (TDG) [68, 69] que segue uma lógica semelhante às tabelas de dispersão distribuídas (TDD) constituídas por (chave, valor), e a função de dispersão fornece a localização para onde devem ser enviados os dados. Os pacotes são assim enviados para o nó destino ou são enviados para o nó mais próximo do nó de destino que o nó de origem alcança.

2.5.2.1 Tabela de dispersão geográfica baseada em células

O uso da TDG permite o envio dos dados para um nó, mas quando a rede estiver densamente povoada, o encaminhamento pode ser mais complicado. Tendo este problema em vista, pensou-se em otimizar o desempenho da tabela de dispersão geográfica aglomerando os nós em células, criando assim uma tabela de dispersão geográfica baseada em células (TDG-C), também conhecida por "*Cell Hash Routing*" [4]. Neste caso não existe a noção de nó mas sim de célula, isto é, a função de dispersão fornece a localização de uma célula e não de um só nó.

Estas células – quadrados de tamanho igual – funcionam como um nó virtual que englobam todos os nós que estão presentes na célula. O tamanho das células é limitado pelo alcance de comunicação dos nós. Estas têm de ser construídas de forma a que, pelo menos, cada nó conheça um dos seus vizinhos. O mais indicado será que todos os nós conheçam todos os seus vizinhos próximos, permitindo a comunicação dos nós dentro da sua célula e com os nós da célula vizinha. Esta característica é importante porque é necessário permitir que exista comunicação entre células, para que exista troca de mensagens na rede.

O encaminhamento mais popular quando se utiliza uma TDG-C é o protocolo de encaminhamento geográfico, "Greedy Perimeter Stateless Routing (GPSR)" [4, 42], podendo apresentar variantes. Este protocolo consiste no envio da mensagem para a célula geograficamente mais próxima da célula de destino. Caso a mensagem chegue a um destino onde não é mais possível o envio para o vizinho mais próximo, o algoritmo retrocede e opta por outro caminho.

Este tipo de solução apresenta a vantagem de criar uma rede estruturada com poucos *clusters*, o que não acontece quando a solução é baseada apenas num nó, sendo os custos inferiores e permite a aplicação de um mecanismo de encaminhamento mais leve. Além disso, suporta entrada e saída de elementos na rede, pois acontece dentro das células, e é mais escalável que a TDG, pois os nós encontram-se dentro das células e estas nunca se alteram.

No sistema proposto, irá ser utilizada uma tabela de dispersão geográfica baseada em células, utilizando as características desta solução para garantir algumas das propriedades

do nosso sistema. Esta solução também permite um encaminhamento utilizando os nós virtuais do qual vamos usufruir utilizando uma variante do GPSR. Nas Secções 3.5 e 3.8 estes assuntos serão abordados com mais detalhe.

2.6 Sumário

Ao longo deste capítulo foram apresentados trabalhos e conceitos relacionados com o trabalho desenvolvido nesta dissertação. O objetivo deste capítulo é enquadrar o leitor nos temas abordados e relacionados com este trabalho, de forma a facilitar o entendimento e compreensão do sistema desenvolvido.

As redes de dispositivos móveis são a base do nosso sistema pois é sobre uma rede deste tipo que o nosso sistema vai operacionalizar. Dessa forma achamos fundamental abordar um pouco o tema, elucidando os leitores sobre esta assunto.

A partilha e armazenamento de dados em redes de dispositivos móveis tem vindo a ser estudada e investigada ao longo do tempo, pois cada vez mais os dispositivos móveis se apresentam mais omnipresentes na sociedade, tornando-se cada vez mais uma fonte de geração e consequente partilha dos dados. Visto isto, apresentamos ao logo do capítulo quais os trabalhos de investigação e quais as aplicações nas lojas de aplicações que mais se relacionam com o nosso sistema. Destes destacamos trabalhos como o Ephseus [79], Krowd [21], iTrust [49] ou MobiTribe [85], onde os autores desenvolveram sistemas com o objetivo de resolver problemas semelhantes ao nosso, mas apresentam limitações que pretendemos resolver como a carga excessiva na rede, a não persistência dos dados, a não resistência a *churn* ou mobilidade, entre outros. Também é importante realçar algumas aplicações de partilha de dados como o *SuperBeam* [48] ou o *Xender* [3] que permitem a comunicação dispositivo-a-dispositivo mas apenas o permitem de um para muitos dispositivos.

Também o modelo de interação publicador/subscritor teve lugar neste capítulo, pois é este o modelo de interação que iremos utilizar no nosso sistema, apesar de um pouco diferente do modelo padrão. No nosso sistema será possívell efetuar subscrições no passado, obtendo dados publicados anteriormente à subscrição, algo que não existe no modelo publicador/subscritor padrão mas, abordado em artigos como no artigo [15].

Por fim, o capítulo terminou com uma abordagem sobre um assunto problemático em sistemas distribuídos: a descoberta e localização dos dados na rede. No final desta abordagem chegamos à conclusão que a utilização de um encaminhamento geográfico combinado com uma tabela de dispersão geográfica baseada em células seria a solução ideal para o nosso sistema.

Em seguida, no Capítulo 3 será apresentado o sistema implementado neste trabalho, o Thyme. Será explicado o seu funcionamento, as suas características, assim como as decisões tomadas na implementação efetuada.

Capítulo 3

Thyme

Neste capítulo é apresentado o sistema implementado, o Thyme. Na Secção 3.1 é apresentada uma visão geral do sistema. Seguidamente na Secção 3.2 é exposta a arquitetura da implementação do sistema e explicados cada um dos seus componentes e as suas interações. Posteriormente, na Secção 3.3 é exposto como a implementação do Thyme pode ser utilizada e quais as operações disponibilizadas. Ao longo deste capítulo são ainda abordados os temas que permitem compreender o Thyme e a sua implementação em redes de dispositivos móveis, como o modelo publicador/subscritor que permite subscrições no passado (Secção 3.4), o armazenamento dos dados (Secção 3.5), a gestão que é feita da grelha da tabela de dispersão geográfica baseada em células (Secção 3.6), a gestão e delimitação do espaço geográfico onde o sistema vai funcionar (secção 3.7), o protocolo de encaminhamento utilizado (Secção 3.8) e como o sistema lida com o movimento dos nós na rede (Secção 3.9). A terminar são apresentados os detalhes de implementação cujo conhecimento é fundamental para a compreensão da implementação efetuada (Secção 3.10). O capítulo termina com uma secção sumária (Secção 3.11).

3.1 O Modelo Thyme

O modelo Thyme apresenta um sistema publicador/subscritor com persistência de dados, desenvolvido para redes de dispositivos móveis. O Thyme permite efetuar as operações típicas de um sistema publicador/subscritor: publicação, subscrição e *dessubscrição*. Além dessas operações também permite a *despublicação* dos dados publicados no sistema, uma vez que o sistema permite um armazenamento persistente dos dados. O subscritor pode ainda obter os dados publicados relativos ao tópico que subscreveu, após ser

notificado da existência desses mesmos objetos.

Os objetos publicados no THYME são identificados para os utilizadores por uma ou mais *tags*, da mesma forma que as *hashtags* nas redes sociais. Dessa forma, quando é efetuada uma publicação, é necessário indicar quais as *tags* que caracterizam o objeto em questão e ao subscrever, o utilizador subscreve sobre uma ou várias *tags* nas quais possui interesse, recebendo notificações dos objetos publicados com a respetiva *tag*.

No THYME, quando é efetuada uma publicação, o sistema guarda o momento em que esta foi efetuada, garantido perceção temporal. Posteriormente, quando for efetuada uma subscrição, esta pode abranger um espaço temporal que pode iniciar e terminar em qualquer momento, inclusive no passado, sendo o utilizador notificado de quaisquer publicações correspondentes à sua subscrição no intervalo de tempo especificado. Esta característica permite ao utilizador obter dados publicados anteriormente à sua subscrição.

Os objetos no THYME são apenas de leitura (*read-only*), não podem nunca ser alterados após serem publicados no sistema. Os utilizadores não podem alterar as publicações que efetuaram, no limite é possível *despublicar* o objeto anterior e publicar um novo objeto com a mesma *tag*, sendo os subscritores dessa *tag* notificados de um novo elemento.

O modelo THYME apresentado em [78] permite a sua materialização utilizando distintas estratégias de armazenamento e encaminhamento: **Publicação local – subscrição global** onde as operações de publicação são efetuadas localmente e as operações de subscrição são enviadas, através de inundação, para todos os elementos da rede, e ***ACD – TDG-C*** que segue uma abordagem de armazenamento centrada nos dados (*ACD*) utilizando uma tabela de dispersão geográfica baseada em células (TDG-C) [4].

A primeira estratégia é adaptada para cenários onde a taxa de mobilidade dos nós é baixa assim como o número de entradas e saídas do sistema. A segunda é mais versátil, pois consegue lidar com os problemas de mobilidade, assim como com a entrada e saída dos nós do sistema, oferecendo também um mecanismo de replicação que permite a persistência dos dados mesmo que o dispositivo publicador tenha saído do sistema.

De acordo com as nossos objetivos, optamos por materializar o THYME utilizando a estratégia ***ACD* – TDG-C**. Optamos por esta estratégia devido à sua resposta perante os problemas de mobilidade e à entrada e saída dos dispositivos na rede mas também devido à sua característica de replicação, tornando os dados persistentes, sendo um dos objetivos do nosso trabalho. Além disso, pensamos que esta estratégia oferece uma maior abrangência a nível aplicacional.

Nesta abordagem, o espaço geográfico onde o sistema está disponível encontra-se dividido em células de tamanho igual, sendo que todos os nós físicos dentro de cada célula funcionam de forma colaborativa representado um nó virtual. Além disso é utilizada uma tabela de dispersão geográfica que indica, através da *tag*, qual a célula responsável pela gestão e armazenamento dos metadados dos objetos publicados e por verificar a correspondência entre subscrições e publicações, com essa *tag*.

Na Figura 3.1 apresentamos um exemplo, utilizando a abordagem ACD - TDG-C, da publicação de uma fotografia com as *tags*: "praia" e "verão". Aplicando a função de

hash conseguimos obter quais as células para onde devemos enviar os metadados pois estas são responsáveis por armazenar e gerir os metadados de cada *tag*, sendo a célula B2 responsável pela *tag* "verão" e a célula C5 responsável pela *tag* "praia". É importante notar que apenas os metadados são enviados para a célula responsável pela *tag*, uma vez que os dados publicados são replicados pelos nós da célula do publicador. Esta estratégia de replicação é denominada de replicação ativa. Além da replicação ativa, o THYME também utiliza os dados obtidos por vontade própria do utilizador, através de obtenção, como réplicas do objeto – denominada replicação passiva.

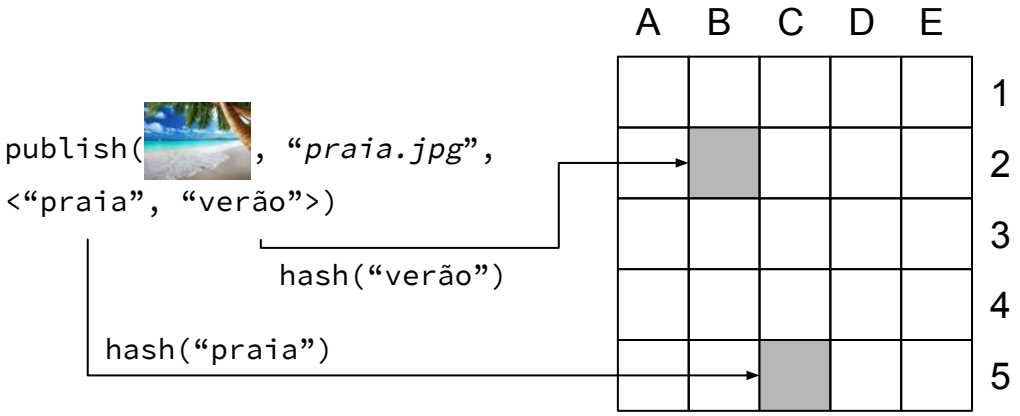


Figura 3.1: Operação de Publicação.

O trabalho realizado nesta dissertação tem como objetivo materializar esta abordagem de forma a ser possível utilizar o THYME em dispositivos móveis com o sistema operativo Android. O modelo THYME encontra-se atualmente implementado no simulador ns-3 [70], utilizando uma rede *ad hoc*, como referido na Secção 1.3. Embora esta seja uma materialização do modelo THYME, dado que as implementações se enquadram em ambientes bastante diferentes, existem novos desafios e adaptações que devem ser tomadas em conta.

Visto que estamos a trabalhar com dispositivos móveis é importante ter em atenção e resolver vários desafios que este contexto acarreta, como os recursos limitados dos dispositivos, a mobilidade inerente a estes dispositivos e a própria comunicação instável. Além disso, em ambientes reais temos de lidar com problemas relacionados com a localização dos nós, o espaço geográfico onde a tabela de dispersão geográfica se enquadra e onde o sistema opera, entre outros desafios.

A implementação deste modelo em dispositivos móveis levou à necessidade de adicionar algumas alterações de forma a resolver os problemas relacionados com o contexto em causa. Uma dessas adições ao modelo foi a noção de "Mundo do THYME". Esta noção representa o espaço geográfico na qual o sistema se encontra operacional e deve ser configurado quando é criada uma nova instância do THYME. Outros pontos importantes a ter em atenção foram o número de mensagens enviadas e recebidas, bem como o seu

tamanho devido à carga na rede que estas acarretariam. O custo energético das operações foi igualmente um ponto sobre o qual nos debruçamos de forma a reduzi-lo, tanto quanto possível. A necessidade da localização dos dispositivos levou à utilização de um mecanismo que permitisse obter essa informação, no nosso caso é utilizado o GPS como mecanismo de obtenção da localização.

3.2 Arquitetura

O THYME é um sistema simétrico, onde cada dispositivo móvel que faz parte do sistema executa a mesma pilha de software de forma independente, como é possível observar na Figura 3.2.

O THYME apresenta uma arquitetura em camadas, constituída por cinco camadas principais, como ilustrado na Figura 3.3. Nesta figura é possível observar camadas com diferentes cores. A diferença de cores é indicativa das camadas desenvolvidas no âmbito desta dissertação e desenvolvidas por terceiros no contexto do projeto *"Hyrax: Crowd-Sourcing mobile devices to develop edge cloud."* [57], onde esta investigação se insere e onde existem equipas a trabalhar em direções de investigação distintas.

Em seguida é apresentada uma descrição *top down* da arquitetura do sistema.

Figura 3.2: Sistema simétrico.

***Interface do* THYME.** A *Interface do* THYME permite a inicialização do sistema e a execução assíncrona das funcionalidades disponíveis pelo sistema. As operações disponíveis pelo THYME são apresentadas na Secção 3.3. Esta interface recorre ao padrão de desenho *facade* de forma a apresentar as funcionalidades dos módulos *Publicador/Subscritor* e *Armazenamento*. Esta camada possibilita a utilização do nosso sistema como uma biblioteca,

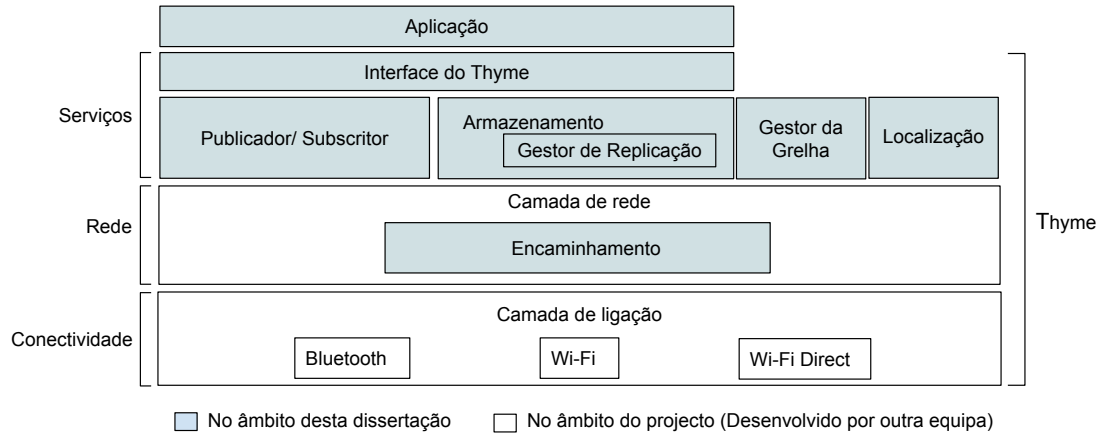


Figura 3.3: Arquitetura do sistema.

podendo ser utilizada nos mais diversos casos de uso, diretamente por uma aplicação ou intermediado por outros sistemas.

Publicador/Subscritor. Este módulo permite a disseminação dos dados entre utilizadores, utilizando um mecanismo de publicador/subscritor com persistência de dados (P/S). O sistema P/S permite o acesso aos dados de um tópico – no nosso caso *tags* – independentemente de quando estes foram publicados, pois no momento da subscrição é possível especificar um intervalo de tempo para o qual a subscrição estará ativa, podendo esse intervalo de tempo incidir no passado, presente ou futuro. As funcionalidades deste módulo foram divididas nas vertentes cliente e servidor, separando claramente a funcionalidade de emitir pedidos ao sistema e tratar os mesmos. O cliente gere os pedidos de publicação e subscrição, bem como a receção das notificações. Por sua vez, o servidor gere o armazenamento e a correspondência entre as subscrições e publicações.

Armazenamento. Este serviço é responsável pelo armazenamento e gestão dos dados armazenados no sistema. É utilizada uma abordagem centrada nos dados, aproveitando as células da TDG, garantindo a replicação ativa dos dados dentro das células. Isto permite que os dados sejam replicados por todos os dispositivos dentro da célula, permitindo assim persistência dos dados. As funcionalidades deste módulo foram divididas nas vertentes cliente e servidor, separando claramente a funcionalidade de emitir pedidos de obtenção de dados ao sistema e a manutenção dos dados armazenados.

Gestor de replicação. A replicação dos dados é monitorizada pelo submódulo *Gestor de replicação*, assegurando que todas as réplicas se mantêm atualizadas e, na nossa implementação, que todos os nós da mesma célula contêm o mesmo estado de replicação podendo responder a pedidos de forma igual, utilizando também as características da TDG baseada em células. Este módulo faz a gestão das réplicas ativas e passivas do nosso sistema.

Gestor da grelha. Este módulo é responsável por dividir e gerir o espaço geográfico divido em células — quadrados de dimensão igual — que funcionam como nós virtuais que representam todos os nós físicos que, num dado instante, se encontram dentro do quadrado associado. Como poderão existir várias grelhas no mesmo espaço geográfico, cada grelha tem um nome e um identificar único associado.

Localização. Serviço responsável por fornecer a localização geográfica do dispositivo ao sistema, assim como fornecer informações sobre o movimento do dispositivo, informando o sistema quando o dispositivo se encontra em movimento e qual a sua posição após parar, pois possivelmente poderá mudar de posição.

Camada de rede. A rede de dispositivos móveis utilizada pelo THYME é criada e suportada por esta camada. Toda a comunicação e formação da rede na nossa implementação do THYME é tratada por uma biblioteca de comunicação, apresentada em [71], desenvolvida por terceiros no contexto do nosso projeto de pesquisa, o *"Hyrax: Crowd-Sourcing mobile devices to develop edge cloud."* [57].

Esta camada é responsável por controlar quais os elementos na rede, permitir a entrada de novos elementos e ainda a função de gerir, manter ativa e funcional a rede de dispositivos móveis. O envio dos pacotes entre vizinhos é efetuado utilizando a camada inferior.

O THYME utiliza esta camada para permitir a comunicação assíncrona entre os vários dispositivos da rede. De forma a receber as mensagens é necessário registar *listeners* sobre as mensagens que são recebidas por esta camada, onde é possível definir qual o tipo de mensagens na qual estamos interessados, recebendo todas as mensagens do tipo definido. O envio de mensagens pode ser direcionado para um nó em específico ou para todos os nós da rede. Esta camada também oferece um serviço de identificação de nós, assunto abordado com maior ênfase na Secção 3.10.8. A camada de rede permite ainda que seja implementado um algoritmo de encaminhamento *multi-hop* a ser utilizado.

Encaminhamento. A camada de rede permite que seja implementado um algoritmo de encaminhamento *multi-hop*, e como será claro na Secção 3.8, optámos por utilizar um algoritmo de encaminhamento geográfico, sendo o módulo *Encaminhamento* a implementação dessa solução.

A camada *Encaminhamento* é responsável por descobrir o caminho que cada pacote deve seguir até ao seu destino, isto é, o caminho desde o nó atual até ao nó de destino. O caminho é descoberto utilizando a posição geográfica dos nós, sendo que cada pacote é enviado para o vizinho mais próximo do nó de destino, dentro do alcance do nó que envia o pacote. De forma a descobrir a localização dos dados é utilizado uma TDG [68].

Camada de ligação. A camada de ligação[1] é responsável pela conexão entre os dispositivos móveis, permitindo a sua comunicação sem fios entre dispositivos vizinhos. Esta conexão pode ser estabelecida utilizando as tecnologias de comunicação existentes nos dispositivos móveis. Neste caso, esta camada está preparada para a comunicação utilizando Wi-Fi, Bluetooth e Wi-Fi Direct.

3.2.1 Interações entre Componentes

O funcionamento do Thyme e a correta realização das suas tarefas apenas é possível devido à interatividade e simbiose entre os diversos componentes. Na Figura 3.4 é possível observar as conexões entre os diversos componentes do Thyme.

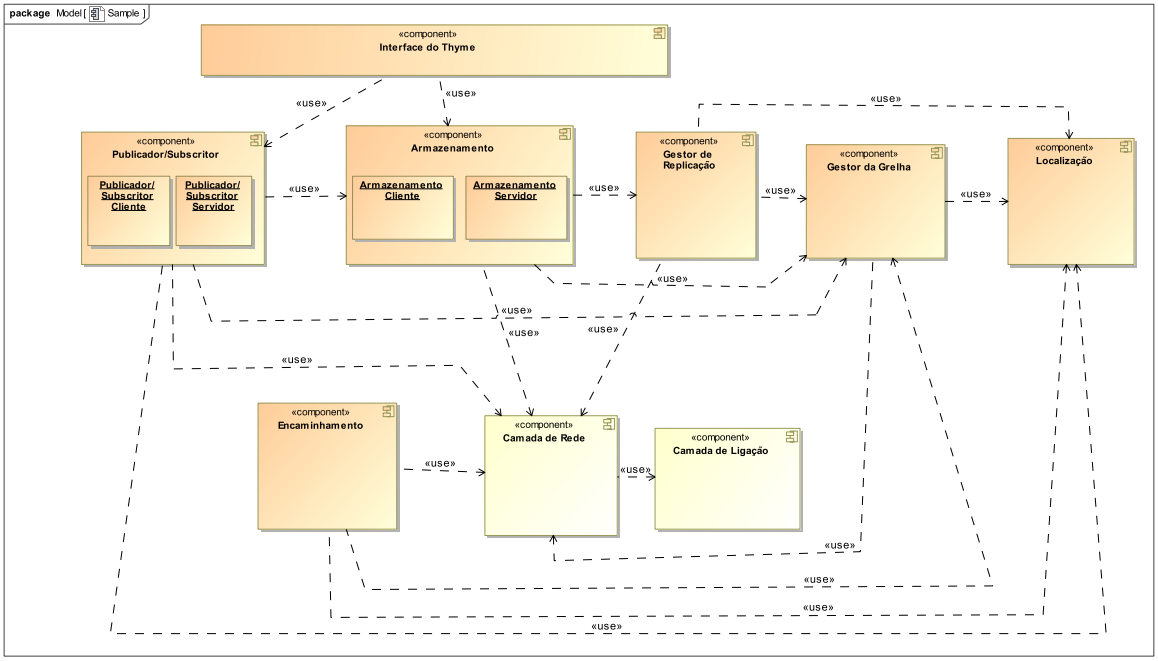


Figura 3.4: Interações entre módulos.

A *Interface do* Thyme é responsável por disponibilizar para os utilizadores os métodos que estes podem executar no sistema, como publicação, subscrição, *despublicação*, *dessubscrição* e obtenção dos objetos publicados, logo existe a necessidade de comunicar diretamente com os módulos responsáveis por essas operações: *Publicador/Subscritor* e *Armazenamento*.

Os dados publicados devem ser armazenados no sistema de forma a poderem ser obtidos posteriormente por outros utilizadores, logo a necessidade da interação entre *Publicador/Subscritor* e *Armazenamento*.

Esse armazenamento dos dados necessita de ser persistente, logo é utilizado o módulo *Gestor de replicação* pelo *Armazenamento* de forma a garantir a replicação dos dados e assim a persistência no sistema.

[1] Esta camada faz parte da biblioteca de comunicação, apresentada em [71].

O *Gestor de grelha* é responsável por gerir todas as questões relacionadas com a divisão do espaço geográfico, coordenar essa informação com as células do sistema, gerir as células e efetuar as operações de distribuição. As informação obtidas deste módulo são necessárias para os módulos *Publicador/Subscritor*, *Armazenamento*, *Gestor de replicação* e *Encaminhamento*.

O módulo de *Localização*, por indicar a posição do dispositivo, é utilizado pelo *Gestor de grelha*, sendo utilizado pelos restantes módulos pela sua capacidade de controlar o movimento do dispositivo.

Por fim, as camadas que controlam a comunicação e a rede são utilizadas pelos restantes módulos, expecto o módulo *Localização*, de forma a permitir a comunicação entre dispositivos.

3.3 Interface do THYME

O THYME pode ser utilizado por terceiros de forma a fazer parte de outro sistema ou ser implementado diretamente numa aplicação. De forma a utilizar o THYME, é necessário inicializar uma instância do THYME criando um objeto desse tipo.

Listagem 3.1: Método *getInstance* do THYME.

```
1 static Thyme getInstance(Activity parent, Context context, Bootstrap bootstrap)
```

A criação de um objeto do tipo THYME é possível através do método estático *getInstance* da classe THYME que retorna uma instância dessa classe, de acordo com os parâmetros de entrada que foram introduzidos. O método *getInstance*, representado na Listagem 3.1, necessita como paramento de entrada a *Activity* da aplicação em que o método foi chamado, o *Context* da aplicação e um objeto que implemente a interface "Bootstrap".

"Bootstrap". A interface "Bootstrap" é um *listener*, que é utilizado para interagir com o utilizador de forma a permitir a inicialização do sistema, sendo fundamental para definir o espaço geográfico no qual o THYME vai operar - a qual denominamos "Mundo do THYME". A criação deste objeto fornece as configurações necessárias para permitir a inicialização do THYME. O "Mundo do THYME" é um assunto de elevada importância para o nosso sistema e por esse motivo será abordada com maior ênfase na Secção 3.7. Nesta secção apenas é utilizado o objeto representativo do "Mundo do THYME" para permitir a inicialização do THYME. Para mais informações sobre o funcionamento e configuração do "Mundo do THYME" deve ser consultada a Secção 3.7.

A interface "Bootstrap" contém quatro métodos, sendo que dois desses métodos necessitam de um retorno pois transmitiram informação para o sistema e dois deles são apenas informativos para a aplicação que estiver a utilizar o sistema. Em seguida é apresentada a assinatura dos métodos da interface "Bootstrap" que devem ser implementados.

selectWorld(existentWorlds) - Quando o sistema inicia, é feita uma procura na rede para perceber se existem outros dispositivos na vizinhança, de forma a saber quais são os "Mundos do THYME" existentes aos quais o novo nó se pode conectar, se assim desejar. Após o fim da procura na rede, este método é chamado, indicando quais os mundos disponíveis, sendo enviado um mapa com o identificador e o nome dos mundos recebidos. Em resposta, a aplicação terá de retornar o identificador do mundo selecionado ou retornará o valor -1 caso pretenda criar um novo mundo. Caso retorne o valor -1, será chamado em seguida o método "onBootstrapFailed();". Caso seja retornado um valor inválido será assumido o comportamento do valor -1.

onBootstrapDone(worldInfo) - No final de todas as tarefas de inicialização este método é chamado de forma a informar que o sistema está pronto a ser utilizado. De forma a fornecer informação é passado como parâmetro dados sobre mundo conectado.

onBootstrapGPS() - Após a definição do mundo, criação de um novo mundo ou conexão a um mundo existente, é necessário descobrir a localização do nó, através do uso do GPS. Este método é utilizado para notificar a aplicação que o sistema se encontra em busca da localização via GPS, o que pode demorar algum tempo, logo será de bom grado avisar o utilizador qual a razão do tempo de espera.

onBootstrapFailed() - Se o nó não encontrar nenhum vizinho após a pesquisa na rede que efetuou ou pretender criar um novo mundo mesmo existindo vizinhos, este método será chamado para informar o utilizador que deve configurar um novo mundo. Como resposta deve ser retornado um objeto da classe "World". Mais informações na Secção 3.7.

Na Listagem 3.2 é possível analisar uma inicialização simples de um objeto "Bootstrap". O método "userSelectReceivedWorld(world)" é apenas representativo, e permite ao utilizador selecionar, ou não, um dos mundos passados em parâmetro. Todo o código utilizado para preencher os métodos é apenas ilustrativo para dar a perceber qual a funcionalidade de cada método.

Inicialização do THYME. Após a definição do "Bootstrap" é possível utiliza-lo para inicializar uma instância do THYME. Na Listagem 3.1 é apresentado como se pode inicializar do sistema. Poderia ser utilizado o objeto "Bootstrap" criado na Listagem 3.2.

Operações. Após a inicialização do THYME é possível utilizar as funcionalidades disponibilizadas pelo sistema, utilizando os métodos presentes na *Interface do* THYME que permite a publicação de objetos, subscrição de *tags*, *despublicação* de objetos publicados, *dessubscrição* de *tags* subscritas e obtenção de objetos publicados sobre *tags* subscritas.

Será importante notar que todos os objetos publicáveis no THYME têm de respeitar uma interface bem definida, denominada "DataItem", de forma a serializar e desserializar

Listagem 3.2: Objecto "Bootstrap".

```
Bootstrap bootstrap = new Bootstrap() {

  @Override
  public byte selectWorld(Map<Byte, String> worlds) {
    return userSelectReceivedWorld(worlds);
  }

  @Override
  public void onBootstrap(String worldName, long end) {
    Toast.makeText(context,
    "Thyme has been initialized with world " + worldName
    + " and its duration is " + end,
    TOAST.LENGTH_LONG).show();
  }

  @Override
  public void onBootstrapGPS() {
    Toast.makeText(context, "Finding your location, please wait.",
    TOAST.LENGTH_LONG).show();
  }

  @Override
  public World failBootstrap() {
    Coordinates coordinates = GPS.getLocation();
    long duration = 7200000; // duracao da instancia do Thyme
    World world = new World(coordinates, 30, 30, 30, 30, "Thyme", endTime);
    return world;
  }

};
```

o objeto em questão. Como referido anteriormente, os objetos no THYME não podem ser alterados, sendo apenas de leitura (*read-only*).

De forma a permitir uma melhor compreensão dos métodos que são fornecidos pelo THYME e facilitando a utilização dos mesmos, será apresentada em seguida uma breve descrição das assinaturas dos métodos.

`publish(dataItem, tags, description, opHandler)`

Este método permite efetuar publicações de objetos no sistema.

`dataItem` – Representação do objeto a ser publicado no sistema.

`tags` – Conjunto de *tags* relacionadas com o objeto a ser publicado.

`description` – Pequena descrição do objeto.

`opHandler` – Implementação do comportamento a ser executado quando a operação terminar, com sucesso ou falha.

`subscribe(tags, startTime, endTime, notHandler, opHandler)`

Método que permite efetuar subscrições no sistema.

`tags` –*Tag* ou conjunto de *tags* a subscrever.

`startTime` – Início do tempo de vida da subscrição.

`endTime` – Fim do tempo de vida da subscrição.

`notHandler` – Implementação do comportamento a ser executado quando é recebida uma notificação relativa a esta subscrição.

`opHandler` – Implementação do comportamento a ser executado quando a operação termina, para o sucesso e para a falha da operação.

`unPublish(objectId, opHandler)`
Com este método é possível remover uma publicação no sistema.

`objectId` – Identificador do objeto a *despublicar*.

`opHandler` – Implementação do comportamento a ser executado quando a operação terminar, com sucesso ou falha.

`unSubscribe(tags, opHandler)`
Este método permite anular subscrições que foram efetuadas no sistema por este dispositivo.

`tags` – Conjunto de *tags* que se pretende anular a subscrição.

`opHandler` – Implementação do comportamento a ser executado quando a operação termina, para o sucesso e para a falha da operação.

`download(metaData, dowHandler)`
Este método permite obter os dados publicados e armazenados no sistema.

`metaData` – Metadados do objeto que se pretende obter.

`dowHandler` – Descrição do comportamento que é executado quando o objeto é recebido e quando a operação falha.

3.3.1 Sequência dos processo das operações

As operações disponíveis pela *Interface do* THYME necessitam da interação entre os vários componentes que constituem a arquitetura do THYME de forma a poderem ser realizadas. Sendo um sistema distribuído, é necessário efetuar comunicação entre os vários dispositivos de forma a conseguir realizar as tarefas pretendidas pelos utilizadores do sistema: publicação, subscrição, apagar subscrição, apagar publicação e obter dados. Nas subsecções seguintes são apresentadas a sequência de ações que são realizadas para efetuar cada operação disponível no THYME.

3.3.1.1 Publicação

A publicação de dados no THYME necessita da interação entre o cliente e o servidor do módulo *Publicador/Subscritor*, podendo ser entre dispositivos distintos. Após a *Interface do* THYME receber a informação que a aplicação pretende fazer uma publicação, é

executado o método responsável pela publicação. Desta forma, o cliente comunica com o servidor, enviando uma mensagem de publicação, onde é enviada toda a informação necessária sobre a publicação. Como resposta, o servidor comunica com o cliente que por sua vez informa a aplicação. É de notar que esta interação é efetuada através de um "*listener*" previamente configurado na aplicação.

Caso existam subscrições ativas interessadas na *tag* correspondente a esta publicação, serão enviadas mensagens de notificações para os respetivos subscritores.

Na Figura 3.5 é possível observar um diagrama de sequência representativo de uma publicação.

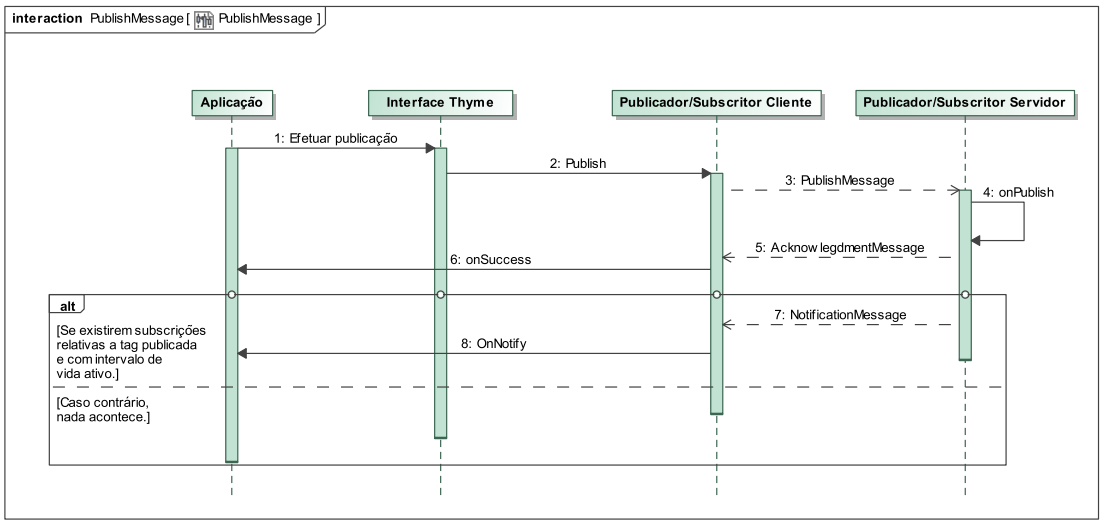


Figura 3.5: Diagrama de sequência da publicação.

3.3.1.2 Remoção da publicação

A remoção de publicação é a operação inversa da publicação e dessa forma a sua sequência de operações é idêntica, sendo diferentes os métodos executados e as mensagens enviadas e recebidas. É possível observar um diagrama de sequência desta operação na Figura 3.6.

3.3.1.3 Subscrição

A subscrição, tal como a publicação, necessita da interação do cliente e servidor do *Publicador/Subscritor*, podendo ser entre diferentes dispositivos, para permitir a sua execução. Tal como na publicação, após a informação da ação de subscrição ser passada da *Interface do* THYME para o cliente, este envia o pedido - via mensagem - para o servidor. Este processa a subscrição, com toda a informação recebida na mensagem, e responde ao cliente com o estado da operação. Na Figura 3.7, é apresentado um diagrama de sequência de uma subscrição onde a subscrição é executada com sucesso.

No caso de existirem publicações relativas à *tag* subscrita e dentro do intervalo subscrito, a aplicação será notificada com uma mensagem de notificação relativa a esses dados.

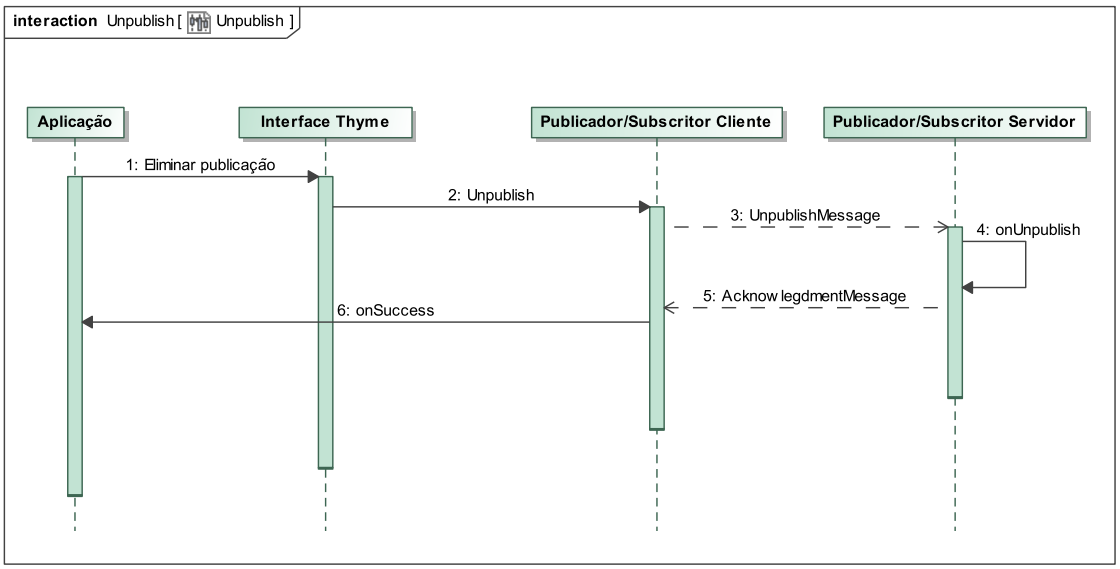


Figura 3.6: Diagrama de sequência da remoção de publicação.

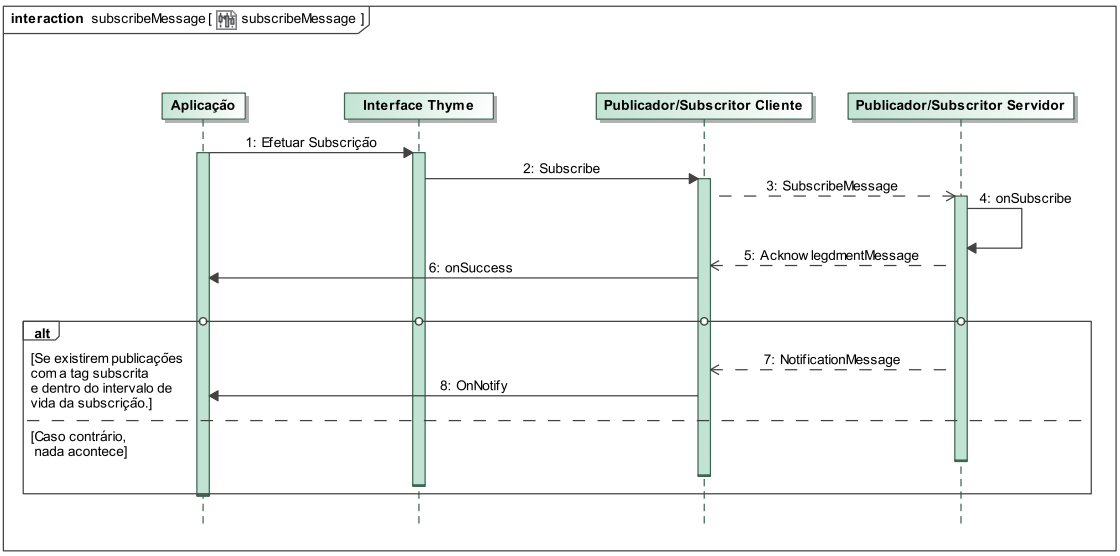


Figura 3.7: Diagrama de sequência da subscrição.

3.3.1.4 Remoção da subscrição

Tratando-se de uma operação que contraria a anterior, a sua sequência de comportamentos no sistema é semelhante à sua inversa - subscrição. Sendo que os métodos e mensagens utilizados são distintos. É possível ver a sequência do comportamento desta operação no diagrama de sequência da Figura 3.8.

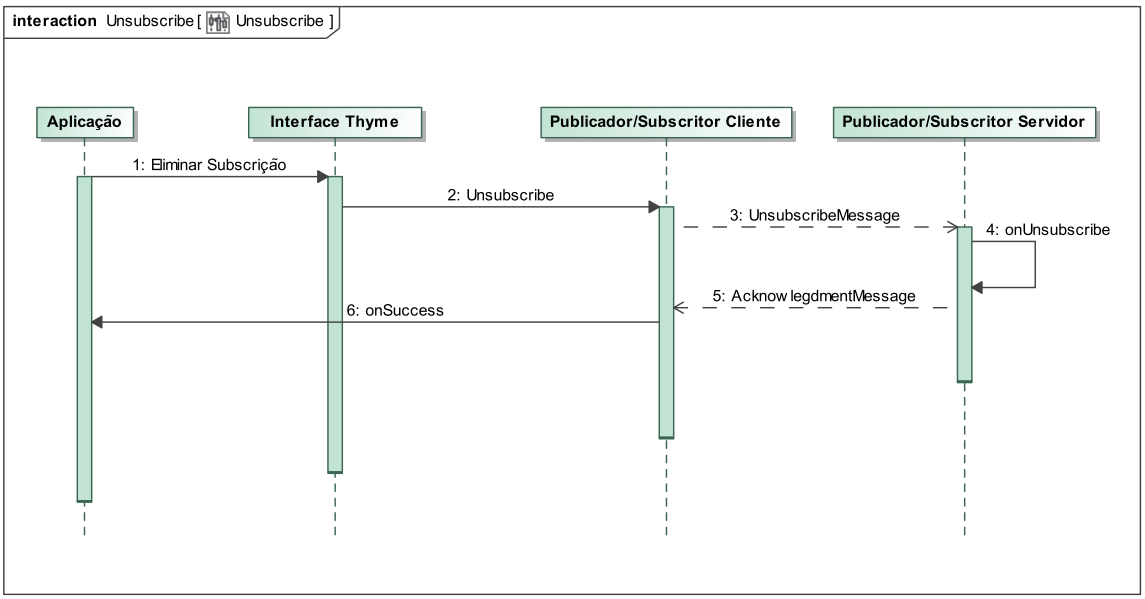


Figura 3.8: Diagrama de sequência da remoção de subscrição.

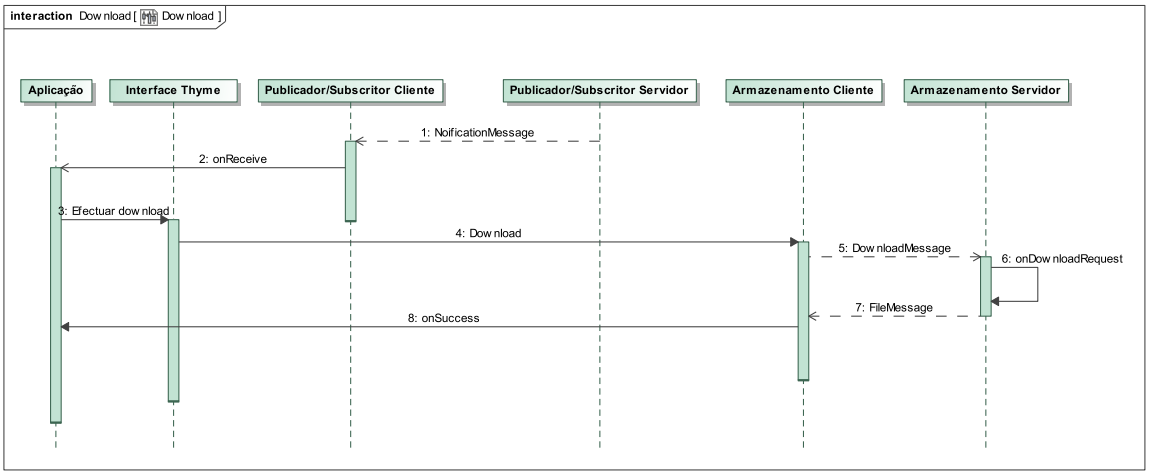


Figura 3.9: Diagrama de sequência da obtenção dos dados.

3.3.1.5 Obtenção dos dados

A obtenção dos dados apenas pode ser efetuada após ser recebida uma notificação sobre esses mesmos dados. Após essa receção pode iniciar-se a operação. Nesta operação é necessária a comunicação entre o cliente do *Publicador/Subscritor*, que controla as notificações, e o cliente e servidor do *Armazenamento* que permite efetuar e responder ao pedido de obtenção dos dados, respetivamente.

Após a notificação e expressa vontade de obtenção desses dados, o cliente do *Armazenamento* informa, através de uma mensagem, o servidor homólogo, possivelmente de outro dispositivo, que pretende obter os dados. Este por sua vez, processa o pedido e responde com uma mensagem que já contém os dados pedidos. Após receber os dados, o cliente tem de informar o sistema que contém esses dados e por fim informa a aplicação

que possui os dados desejados. Tal como na publicação e subscrição, esta interação não é efetuada diretamente, é efetuada através de um *"listener"* previamente configurado na aplicação.

3.4 Publicador/Subscritor

O Thyme utiliza um modelo Publicador/Subscritor que podemos considerar persistente pois é possível obter os objetos publicados no sistema no futuro, no presente e no passado. Os dados são armazenados utilizando uma abordagem de armazenamento centrada em dados utilizando a tabela de dispersão geográfica baseada em células. Na nossa abordagem o modelo publicador/subscritor utiliza a noção de célula transmitida pela TDG baseada em células.

Devido ao contexto em que este trabalho foi desenvolvido tentamos utilizar o menor número de mensagens enviadas pela rede, assim como o menor tamanho das mesmas, isto para tentar reduzir o número de colisões de mensagens, diminuindo as interferências, e para tentar alcançar um baixo consumo de energia, visto ser necessário preservar o tempo de vida das baterias dos dispositivos móveis.

Em seguida iremos explicar as características e como funcionam as operações disponíveis por este modelo publicador/subscritor com persistência.

3.4.1 Publicação de dados

No Thyme, o objeto é a unidade básica de trabalho e é visto como um conjunto opaco de bytes. A cada objeto está associado um conjunto de metadados que consistem no tuplo:

$$\langle id_{\text{obj}}, T, s, ts_{\text{pub}}, id_{\text{prop}}, rep \rangle$$

id_{obj} - é o identificador do objeto (conjunto de bytes);

T - representa o conjunto de *tags* relacionadas com o objeto (similar às *hastags* usadas nas redes sociais);

s - uma pequena descrição do objeto (uma miniatura no caso das imagens);

ts_{pub} - tempo de publicação;

id_{prop} - identificador do nó proprietário do objeto; e

rep - localização das réplicas do objeto.

A chave do objeto dentro do sistema é um par $\langle id_{\text{obj}}, id_{\text{prop}} \rangle$, onde id_{obj} é o identificador do objeto e id_{prop} é o identificador do seu proprietário, isto é, do nó que publicou esse objeto. Isto garante que cada objeto é único dentro do sistema. Se dois nós publicarem

conteúdo com o mesmo identificador, nunca serão considerados iguais pois o segundo elemento do par servirá como diferenciador.

Quando é executada uma operação de publicação, é efetuada uma indexação dos metadados do objeto pelas suas *tags*, criando assim um índice por *tag*. Isto significa que os metadados são enviados para a célula indexada pelas *tag(s)* respetiva(s) mas o objeto em si é replicado apenas na sua célula de origem. Este índice permite que o objeto (potencialmente grande) não navegue na rede, pois fica armazenado na célula de origem, e que sejam apenas os metadados (provavelmente mais pequenos) a serem transmitidos, diminuindo assim o tráfego na rede. Por sua vez, o índice por *tag* também é utilizado para verificar se as subscrições correspondem às publicações que já foram efetuadas, conseguindo assim comparar esta informação com o mínimo de mensagens na rede. Quando existe mais que uma *tag* associada ao objeto, é necessário efetuar tantas indexações quanto o número de *tags* correspondentes.

3.4.1.1 Remoção de dados

No THYME é possível remover os dados que foram publicados. Isso é praticável através da operação inversa da publicação, a *despublicação*, onde os passos são os mesmos que a publicação mas com o objetivo de revogar a publicação. Esta operação torna o objeto inacessível a subscrições futuras, mesmo que sejam subscrições no passado.

Os metadados do objeto a *despublicar* são eliminados das células responsáveis pelas suas *tags*, isto é, dos nós da célula indexada pelas *tags* do objeto em questão. Em relação ao objeto, nas réplicas ativas é eliminado mas nas réplicas passivas não são eliminadas pois são cópias obtidas por descarregamento explícito logo continuarão a existir, no entanto os dados *despublicados* ficarão inacessíveis nesses nós, pois não existem os metadados do objeto, logo não é possível obtê-lo. Mas se existir algum dispositivo que tenha recebido uma notificação antes da *despublicação*, ainda poderá obter os dados, acedendo ao objeto nas replicas passivas. Em ultimo caso o objeto será obtido no próprio publicador, uma vez que poderá não existir mais replicas passivas e o publicador é considerado uma réplica passiva pois contêm o objeto por vontade própria.

3.4.2 Subscrição

Sendo o tempo uma dimensão de primeira ordem na disseminação de dados no THYME, a subscrição é composta por duas partes: a consulta (ou filtro) — uma fórmula de proposição lógica composta por *tags* e operadores lógicos, e o intervalo de vida da subscrição (início e fim). Atualmente na implementação apresentada, não é suportada uma fórmula de proposição lógica mas sim um conjunto de *tags* a serem subscritas.

Mais especificamente, no THYME a subscrição consiste num tuplo:

$$\langle id_{\text{sub}}, q, ts^{i}, ts^{f}, id_{\text{prop}}, cell_{\text{prop}} \rangle$$

onde,

id_{sub} é o identificador da subscrição;

q representa a consulta que define quais os eventos relevantes para a subscrição, sendo um conjunto de *tags*;

ts^i é o tempo de inicio da subscrição;

ts^f é o tempo de termino da subscrição;

id_{prop} identificador do nó subscritor; e

$cell_{\text{prop}}$ identificador da célula do nó subscritor.

A consulta da subscrição trata-se de uma fórmula em lógica proposicional onde as *tags* são associadas aos dados publicados. A $cell_{\text{prop}}$ é necessária pois o encaminhamento geográfico apenas conhece o posicionamento geográfico das células, logo o envio de notificações é feito para a célula do subscritor e internamente a mensagem será encaminhada para o nó correspondente. Esta informação necessita de ser atualizada sempre que o nó subscritor mudar de célula. As estampilhas temporais ts^i e ts^f, são usadas para controlar o intervalo de tempo em que a subscrição é válida. Todos os eventos começam no instante 0 (zero) mas podem ter durações diferentes. Sendo que o utilizador pode querer abranger parte ou todo o evento.

A subscrição no THYME consiste no registo do nó subscritor na célula responsável pela chave da função de dispersão da respetiva *tag*. Este método permite uma fácil verificação de correspondência entre publicações e subscrições através de uma pesquisa simples, uma vez que os metadados são indexados por *tags*. Caso haja correspondência serão enviadas notificações informando os nós subscritores dos objetos em questão.

3.4.2.1 Remoção da subscrição

Após uma subscrição, é possível efetuar a operação contrária, *dessubscrição*, de forma a não receber mais notificações sobre as respetivas *tags*. A *dessubscrição* consiste no envio de uma mensagem para a célula (ou células) determinada pela função de *hash* de acordo com a *tag* a *desubscrever*. Os elementos da célula responsável ao receberem essa mensagem irão eliminar o nó remetente da lista de subscritores da respetiva *tag*, deixando assim de receber as notificações desse evento.

3.4.2.2 Notificação

A notificação enviada pelo sistema tem a finalidade de informar os subscritores da existência de objetos publicados no sistema, relativos ao evento subscrito, neste caso, relativos à *tag* subscrita.

As notificações podem ser enviadas em duas situações diferentes: 1) quando é efetuada a própria subscrição e 2) quando é efetuada uma publicação relativa à *tag* subscrita.

Quando é efetuada uma subscrição, a informação relativa à subscrição é enviada para a célula indexada por *tag*, sendo essa célula responsável por verificar se existe alguma publicação que corresponda à nova subscrição. Visto que a subscrição apresenta um tempo de vida que pode ser anterior à própria subscrição - ts^i - a verificação permite averiguar se existe alguma publicação no passado, de acordo com o tempo de vida da subscrição, conseguindo assim notificar os utilizadores de publicações anteriores à sua subscrição. Como as subscrições são armazenadas num mapa indexado por *tags* o custo desta pesquisa é baixo.

Ao ser efetuada uma publicação, os metadados do objeto são indexados por *tag* e enviados para a célula responsável pela respetiva *tag*. Após a receção desses novos metadados pela célula indicada através da função de *hash*, é efectuada uma verificação das subscrições armazenadas localmente para encontrar, se existir, subscrições relativas a essa *tag*. Essa verificação, tem em conta que o tempo de vida da subscrição tem de ser válido, isto é, inferior ao tempo definido como fim da atividade da subscrição - ts^f. Ao enviar estas notificações o sistema está a informar o subscritor de subscrições no presente.

De forma a minimizar a informação transmitida na rede, as notificações enviadas para os nós subscritores contêm apenas os metadados e não os respetivos dados, sendo que os dados podem ser obtidos posteriormente após o conhecimento dos metadados. Os metadados são o mínimo de informação necessária para permitir o descarregamento dos objetos.

3.4.3 Obtenção dos dados

A descoberta da localização dos objetos é um problema clássico em sistemas distribuídos, sendo um problema mais complexo quando estamos perante um contexto onde são usados dispositivos móveis, pois a mobilidade dos nós é uma característica implícita.

De forma a resolver este problema, o THYME utiliza as características da TDG baseada em células, onde indexamos os objetos pela sua *tag* sendo a célula indexada responsável por gerir e armazenar os metadados. Nestes metadados estão presentes as informações necessárias para a obtenção do objeto - identificador do objeto e *tag* correspondente - e a lista de réplicas do objeto, onde será possível obter o mesmo. Aparentemente esta solução obrigaria à necessidade de duas operações/comunicações para obter o objeto: em primeiro lugar seria necessário comunicar com a célula responsável pelos metadados de forma a obter os metadados e em seguida comunicar com um nó que contenha a réplica do objeto de forma a obter o mesmo. Na verdade, seriam necessárias essas duas operações/comunicações, mas na implementação do THYME optamos por apenas permitir a obtenção dos dados após receber uma notificação. Assim, sendo que cada notificação transporta os metadados do objeto e estes metadados contêm uma lista de todas as réplicas onde é possível obter o objeto, $list_{\text{rep}}$, torna-se possível obter o objeto efetuando apenas uma operação - comunicação com uma das réplicas que contém o objeto.

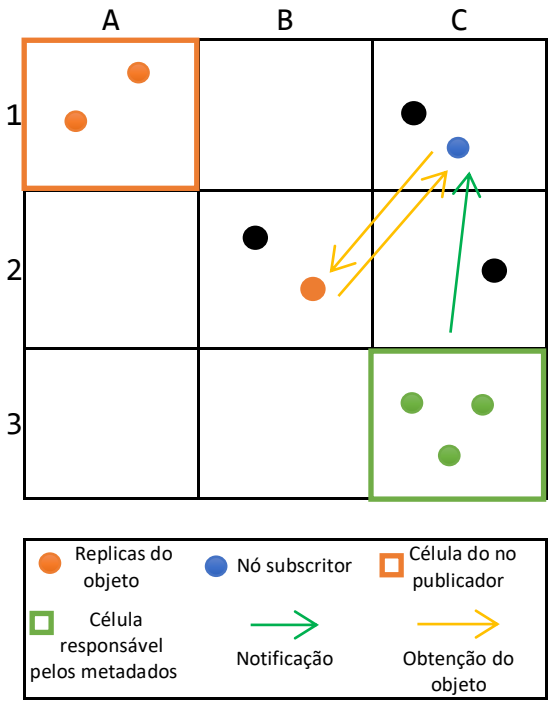


Figura 3.10: Obtenção dos dados.

Visto que a lista de localizações contém todas as réplicas do objeto, utilizando novamente as características da TDG baseada em células e o facto de os dispositivos móveis conseguirem identificar a sua posição, é escolhida a réplica geograficamente mais próxima do dispositivo que pretende obter o objeto. Esta escolha permite reduzir a distância que as mensagens têm de percorrer na rede, sendo mais relevante neste caso pois o objeto é, provavelmente, uma quantidade elevada de dados sendo que a sua mensagem será bastante pesada. Optar pela réplica mais próxima permite assim reduzir a distância percorrida na rede, o tempo de transmissão e a carga na rede.

Na Figura 3.10[2] encontra-se representado um processo de obtenção de dados. Neste caso o nó subscritor, representado a azul, recebe uma notificação enviada pela célula responsável pela gestão dos metadados e das subscrições, célula verde, após uma publicação ou após a própria subscrição, se o tempo de vida da subscrição abranger o passado e existir dados relevantes. Após receber a notificação, o nó pretende obter o objeto notificado e para isso opta pela réplica com a localização na célula B2, pois encontra-se geometricamente mais próximo diminuindo assim a distância percorrida na rede quer do pedido quer da resposta, que pode ser pesada devido ao tamanho, provavelmente, elevado do

[2] A representação usada para identificar as células é apenas para facilitar a compreensão pois no sistema não é usado este tipo de identificação.

objeto.

Devido às características dos dispositivos móveis, baixa durabilidade das baterias e elevada mobilidade, decidimos adicionar um tempo de vida ao pedido de obtenção dos dados, isto porque pode ser efetuado um pedido a uma réplica, mas esta pode ter saído do sistema. No final deste tempo de vida, se não existir resposta por parte da réplica escolhida, esse pedido será cancelado e será escolhida outra réplica - a réplica mais próxima excluindo a anterior - e será enviado um novo pedido. Este ciclo de operações será executado três vezes (valor configurável), se ao final da terceira tentativa não existir resposta informamos o subscritor que o objeto em questão não existe no sistema.

3.5 Armazenamento

Tendo em conta o ambiente volátil e dinâmico para o qual este sistema está direcionado, é importante garantir (tanto quanto possível) que os dados que são publicados assim como os dados do sistema não desapareçam ou fiquem inacessíveis. De forma a assegurar a persistência dos dados, a disponibilidade dos conteúdos e a tolerância à entrada e saída de dispositivos (*churn*), são utilizadas algumas técnicas para permitir o armazenamento persistente dos dados.

Na nossa abordagem do THYME é utilizada uma tabela de dispersão geográfica baseada em células para garantir o armazenamento persistente dos dados do sistema, utilizando uma abordagem de armazenamento centrado em dados (ACD) [68]. Esta abordagem permite que os dados, sejam armazenados por um identificador específico, no nossa caso serão as *tags*, o que permite que a localização dos dados seja obtida através da *tag* respetiva [40]. Com esta abordagem garante-se a vantagem de ser possível a consulta da localização dos dados através de uma *tag* correspondente a esses dados evitando a necessidade de efetuar uma inundação na rede, o que poderia introduzir bastante carga na rede. A utilização desta solução leva a que o armazenamento dos dados seja local à célula, sendo que é utilizada pelo THYME para armazenar os metadados dos objetos publicados assim como as subscrições.

Em relação aos dados publicados, o THYME faz uso de duas estratégias de replicação: *Replicação ativa* - replicação efetuada de forma ativa pelo sistema, dentro das células do nó de origem; e *Replicação passiva* - consiste em utilizar os nós que obtiveram esses dados, por vontade própria, como réplicas.

A replicação dos dados publicados apenas é vantajosa, se o sistema tiver conhecimento da localização das réplicas de cada objeto. De maneira a garantir essa informação, existe nos metadados um campo responsável pelas réplicas dos objetos: *rep*. Esta informação permite garantir o armazenamento das várias localizações, tornando-se uma lista de localizações ($list_{\text{rep}}$) onde estão armazenadas todas as localizações das réplicas do objeto correspondente a esses metadados.

Replicação Ativa. A replicação ativa consiste na replicação dos dados publicados dentro da célula do seu publicador. Utilizando as características dos nós virtuais das TDG baseada em células, quando é efetuada uma publicação, o objeto em questão é replicado em todos os nós da sua célula. A Figura 3.11 representa o processo de replicação ativa, sendo que o nó laranja (na célula B2) após publicar um objeto, esse objeto é replicado em todos os nós dessa célula sendo que nó final todos eles contem o objeto em questão.

Uma vez que a chave da função de dispersão corresponde a uma célula, esta replicação permite que todos os nós da célula tenham a capacidade de responder a pedidos desse objeto. Este mecanismo garante tolerância à entrada e saída de dispositivos, garantindo-se persistência e disponibilidade dos dados, pois se o nó publicador sair da rede o objeto irá manter-se na rede. Além disso contribui para diminuição do número e do tamanho das mensagens a circular na rede, dado que o envio de dados pesados apenas acontece dentro da célula e não entre células.

Na lista de localizações ($list_{\text{rep}}$) é registado um identificador especial, de forma a indicar qual a célula de origem, garantindo a existência do objeto nessa localização, ou seja na célula, mesmo que o nó publicador tenha abandonado o sistema.

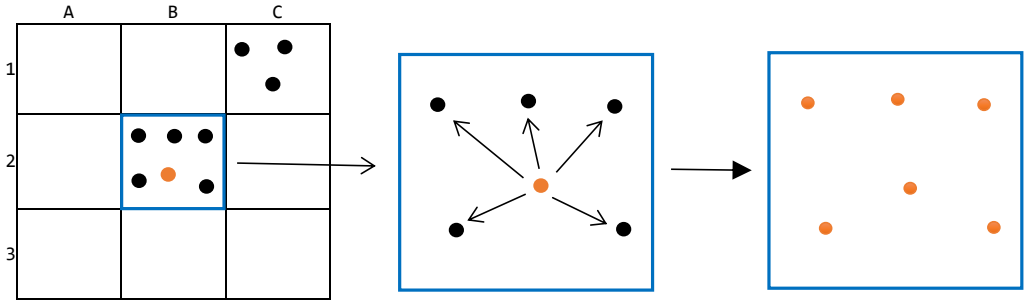


Figura 3.11: Replicação ativa.

Replicação Passiva. A replicação passiva tira partido das cópias do objeto dispersas pelos vários nós/células do sistema, que foram obtidas por descarregamento explícito. A Figura 3.12 representa o processo de replicação passiva, onde um nó da célula C1 obtém o objeto publicado por um nó da célula B2, sendo que após essa obtenção o próprio nó passa a ser uma réplica do objeto, sendo uma réplica passiva pois o obtenção do objeto foi por vontade própria e não imposta pelo sistema.

Este mecanismo de replicação oferece duas vantagens ao sistema: por um lado concede mais disponibilidade dos dados, pois existem mais réplicas dos objetos; e por outro garante uma maior dispersão dos conteúdos na rede, podendo os nós interessados optar pela réplica mais próxima, diminuindo assim a distância percorrida pelas mensagens

na rede. Como os nós são voláteis, se estes se movimentarem é necessário atualizar a localização das réplicas passivas após a estabilização do nó.

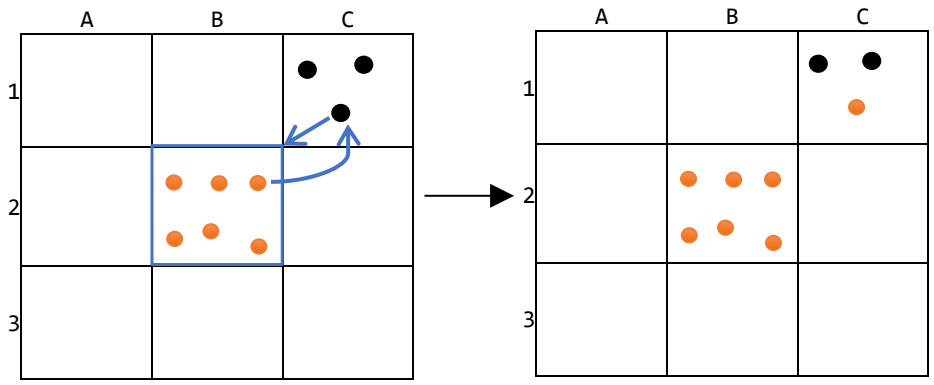


Figura 3.12: Replicação passiva.

Na lista de localizações ($list_{\text{rep}}$) é adicionada a localização do objeto após este efetuar a obtenção do objeto.

3.6 Gestão da Grelha

Como referido anteriormente no Capítulo 3.5, o armazenamento dos dados no THYME é possível utilizando tabela de dispersão geográfica baseada em células para permitir o armazenamento persiste dos dados no sistema. Dessa forma é importante perceber como é feita a gestão da grelha e como funciona esta TDG-C.

A tabela de dispersão geográfica (TDG) [68] utilizada é semelhante a uma tabela de dispersão comum, onde os nós têm a noção de posição geográfica e cada chave corresponde a uma posição geográfica. Ao efetuar a função de distribuição, o nó responsável pela chave correspondente é o nó mais próximo da posição geográfica resultado da função de distribuição efetuada.

Após analisarmos o comportamento da TDG, observamos que o seu funcionamento em redes densas seria um pouco limitado pois o encaminhamento necessitaria de ser efetuado utilizando vários nós o que seria difícil de gerir. De forma a colmatar este problema e com base em trabalhos como [46], [60], [75] e principalmente [4], decidimos optar pela utilização de uma TDG mas utilizando um paradigma baseado em células (TDG-C) [4].

Com esta solução, o espaço geográfico é dividido em células - quadrados de tamanho igual - sendo que todos os nós presentes nessa célula comunicam em simbiose, de forma a ser possível que o sistema trate cada célula como um nó virtual. Esta divisão em células permite que todos os nós mantenham o mesmo estado, garantindo que todos os nós

possam responder com a mesma informação quando for efetuado um pedido para a sua célula.

Utilizando uma TDG-C a função de dispersão deixará de corresponder a um nó passando a identificar uma célula, sendo as células (nó virtual) responsáveis pelas chaves e não um nó físico. Quando é necessário enviar uma mensagem, é efetuada a função de distribuição com a chave correspondente, no nosso caso *tags*, sendo o resultado dessa função uma célula, logo a mensagem é enviada para essa célula - nó virtual. Na célula correspondente, um nó aleatório vai receber a mensagem e vai enviar essa mensagem para todos os nós da sua célula. Todos os nós processam a operação em questão mantendo-se todos os nós com o mesmo estado, garantindo assim a coerência do nó virtual. No entanto apenas um dos nós, o que recebeu a mensagem inicialmente, irá responder ao nó que efetuou o pedido. Na Figura 3.13 é possível observar uma representação do funcionamento dos nós virtuais.

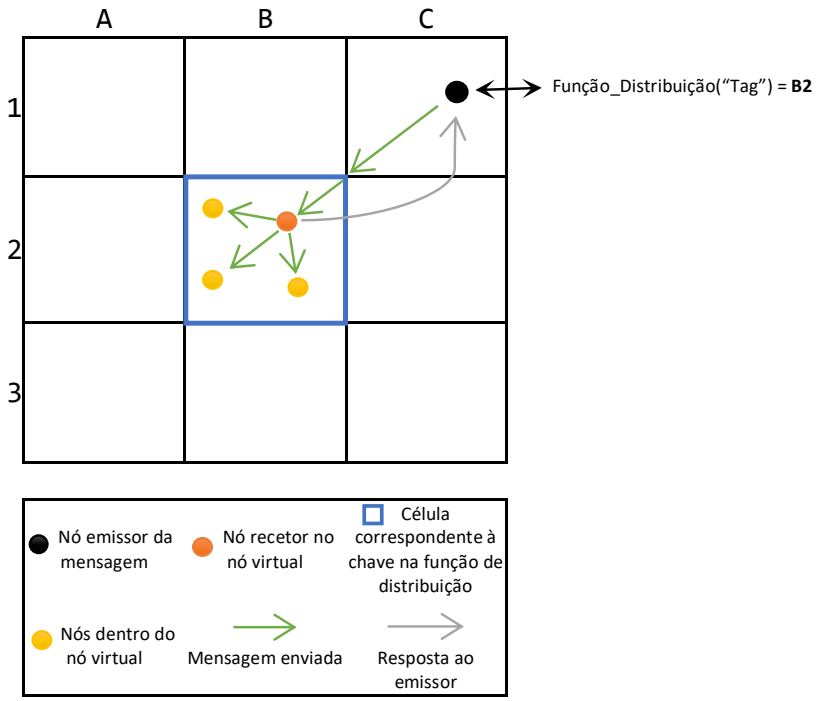


Figura 3.13: Nó virtual.

De forma a garantir o funcionamento desta solução, é necessário que o tamanho das células possibilite que os nós de uma célula, conheçam os restantes nós dessa célula e pelo menos um nó das células adjacentes povoadas. Esta exigência permite que exista sempre comunicação entre células e permite que todos os nós dentro da célula consigam colaborar de forma a manter o mesmo estado dentro da célula.

3.6.1 Novo nó

Quando um novo nó se junta a uma célula é necessário atualizar esse novo nó de forma a que este passe a possuir o mesmo estado que os restantes nós da sua célula. Desta forma garantimos que todos os nós contêm o mesmo estado, mantendo as características da TDG-C, podendo assim responder a pedidos direcionados para a sua célula com o mesmo estado que os restantes.

A atualização do nó na sua nova célula é efetuada utilizando o estado dos seus vizinhos na mesma célula. Quando o nó entra na nova célula, envia uma mensagem para um dos nós previamente existente nessa célula, de forma a pedir o estado da célula para que este possa fazer parte do nó virtual. De forma a prevenir falhas ou atrasos na comunicação, o nó envia a mensagem e espera 5 segundos, valor configurável. Se não obtiver resposta, volta a tentar efetuar a comunicação mas desta vez para outro vizinho, se existir, sendo que este mecanismo se repete três vezes, em caso de falha. Se ao fim da terceira vez não existirem vizinhos, o nó assume que é o primeiro nessa célula.

No exemplo da Figura 3.14, o novo nó da célula tem vizinhos, por isso envia a mensagem para um dos vizinhos de forma a obter o estado da célula. Esse nó ao receber a mensagem responde ao novo nó com o estado atual da célula: metadados armazenados, subscrições armazenadas e objetos publicados por elementos da célula – replicação ativa. Ao receber a resposta ao seu pedido, o novo nó atualiza-se com toda a informação e pode agora começar a participar como um nó ativo na sua nova célula.

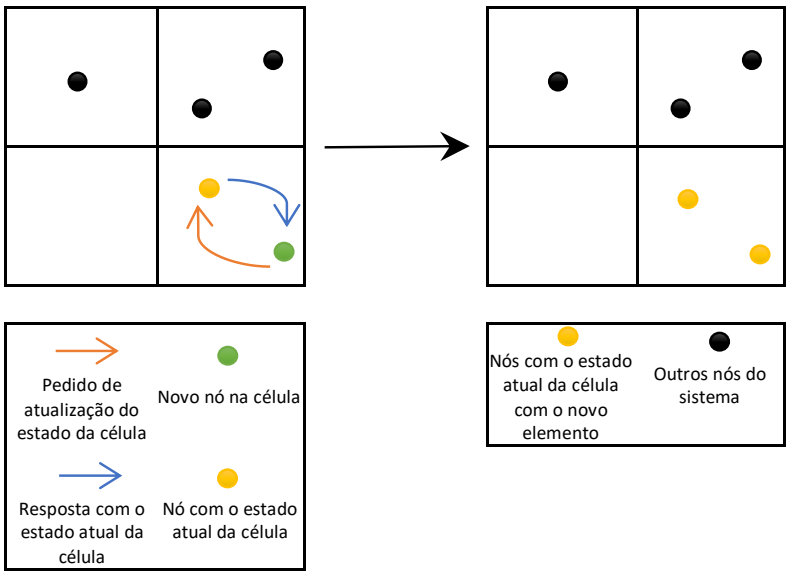


Figura 3.14: Entrada de um novo nó.

3.7 Mundo do Thyme

A utilização da TDG-C necessita que o espaço geográfico seja configurado inicialmente indicando qual é o espaço geográfico que deve ser considerado na divisão das células. Isto permite que a divisão das células considere apenas espaço geográfico pretendido, evitando que seja possível a seleção de células onde não existe conexão ou demasiado distantes, tornando o sistema disfuncional. De forma a evitar este problema, é definido inicialmente o espaço geográfico abrangido pela TDG-C e onde o Thyme opera, a qual denominamos "Mundo do Thyme".

O "Mundo do Thyme" contém a informação necessária para definir o espaço geográfico, e permite definir qual o tipo de comunicação que pode ser efetuada entre os restantes nós, isto é, apenas os dados publicados por nós do mesmo mundo podem ser obtidos por elementos desse mundo. Isto porque dentro do mesmo espaço geográfico podem existir vários "Mundos do Thyme" mas apenas os nós com o mesmo mundo podem obter dados desse mundo, mas a nível de persistência dos dados todos os nós dentro do espaço geográfico podem contribuir. Na Figura 3.15 apresentamos um exemplo que pretende explicar melhor esta situação.

Toda a tabela da Figura 3.15 corresponde ao espaço geográfico disponível. O quadrado amarelo e o quadrado verde são dois "Mundos do Thyme" distintos e os nós amarelos e verdes correspondem aos "Mundos do Thyme" definidos por essas cores. Se o nó verde com contorno preto pretender fazer uma operação, publicação por exemplo, o resultado da função de distribuição da *tag* será obrigatoriamente dentro do quadrado verde, pois são esses os limites do mundo definido. Supondo que a função de dispersão selecionou a célula D3, neste caso o pedido será enviado para essa célula e todos os nós da célula, independentemente da cor, vão receber a mensagem e vão armazenar esses metadados, podendo no limite ser um nó amarelo a receber o pedido e a responder ao nó verde, mas o nó verde nunca poderá obter dados publicados por nós amarelos e os nós amarelos nunca poderão obter dados publicados por nós verdes, apesar de todos os nós contribuírem para o armazenamento dos metadados. Também a nível da replicação ativa dos dados, todos os nós receberão uma cópia dos dados, na célula do publicador - D4, mas apenas poderão enviar esses dados para nós da cor verde.

Quando o Thyme inicia existem duas opções para a definição do "Mundo":

- conexão a um mundo já existente; ou
- criação de um novo mundo, podendo existir um número indeterminado de mundos.

Quando a decisão é a criação de um novo mundo, pois pode não existir nenhum outro mundo ou por outro motivo, é necessário configurar o mundo de acordo com a definição do Mundo em 3.7. Após terminar toda a configuração necessária, como a divisão do espaço geográfico em células – abordado em 3.7 –, a descoberta da sua célula, entre outras, o nó inicia o envio de mensagens periódicas indicando aos restantes vizinhos, se existirem, que

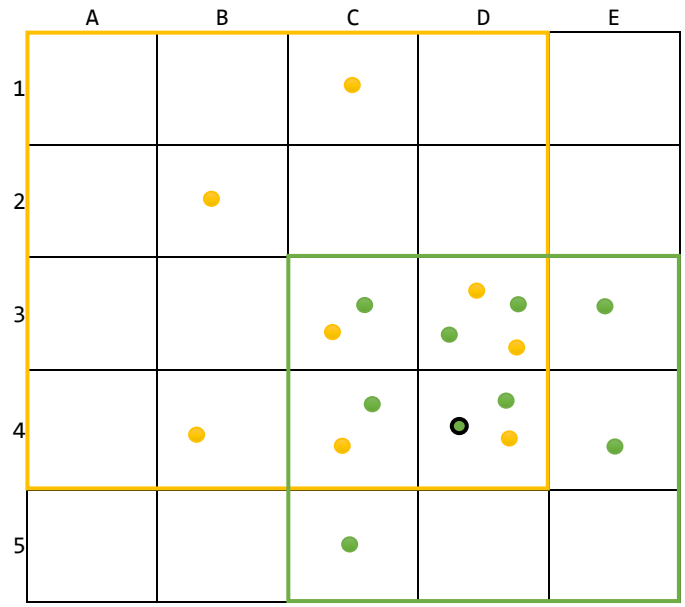


Figura 3.15: Vários "Mundos do Thyme".

está vivo e encontra-se no sistema, mas também para informar aos novos nós que o seu mundo existe e podem conectar-se a ele. Estas mensagens são denominadas pelo sistema de mensagens "Hello".

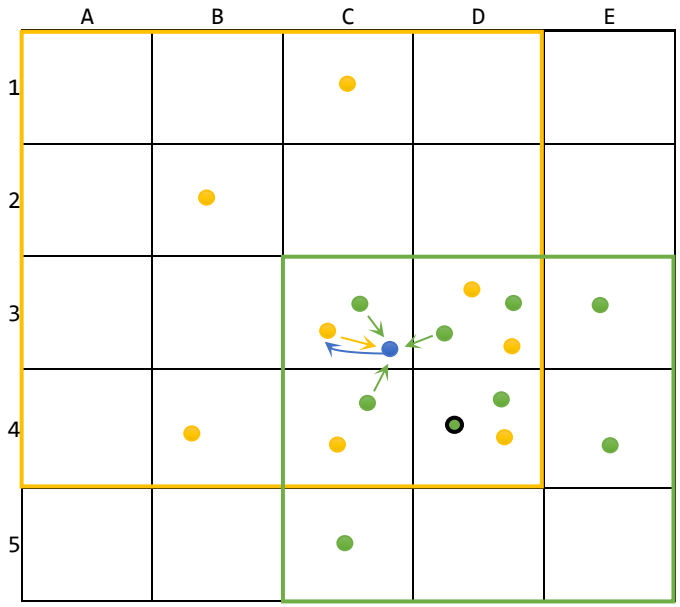


Figura 3.16: Seleção entre dois "Mundos do Thyme".

No caso anterior, na existência de dois mundos, e acompanhando a Figura 3.16, um novo elemento surge no espaço geográfico onde alcança os dois mundos, nó azul. Neste caso, o sistema recebe as mensagens periódicas dos restantes nós e indica quais os mundos disponíveis e a qual este novo nó se pode ligar mas também possibilita a criação de um novo mundo. Supondo que o nó azul pretende ligar-se ao mundo delimitado pela cor amarela, será enviada uma mensagem ao nó amarelo mais próximo indicando que se pretende ligar ao seu mundo, pedido as informação necessárias para a configuração do mesmo. Após receber a mensagem com o "Mundo do Thyme" correspondente ao mundo amarelo, o nó azul efetua as configurações necessárias, como a configuração das células e o cálculo da sua posição no mundo, isto é, a sua célula, e após essa configuração e após o novo se atualizar com os dados do nó virtual, este pode começar a comunicar com restantes nós do seu mundo.

Configuração do espaço geográfico. Quando se decide criar um novo "Mundo do Thyme" é necessário introduzir algumas informações fundamentais para a definição do mesmo. Algumas dessas definições estão representadas na Figura 3.17, sendo apresentadas todas as informações necessárias em seguida:

- Centro do mundo - Ponto de referência do Mundo, normalmente é a localização do nó .
- Norte - Quantos metros a norte são abrangidos pelo "Mundo do Thyme".
- Sul - Quantos metros a sul são abrangidos pelo "Mundo do Thyme".
- Este - Quantos metros a este são abrangidos pelo "Mundo do Thyme".
- Oeste - Quantos metros a oeste são abrangidos pelo "Mundo do Thyme".
- Nome - O nome identificador do "Mundo do Thyme", será útil para os identificar e escolher entre os mundos existentes.
- Duração - Define a duração do novo mundo, em milissegundos, pois pode existir a necessidade da criação de mundos de curta duração ou longa duração dependendo da finalidade.

Além destas informações, o "Mundo do Thyme " contém ainda mais duas informações bastante relevantes: o identificador do mundo e o tamanho das células.

Cada mundo contém um identificador único. Este identificador é utilizado para verificar se o nó que efetuou o pedido pertence ao mesmo mundo dos dados que pretende obter, caso contrário não serão enviados.

A duração do mundo, como referido anteriormente, é o tempo de vida do "Mundo do Thyme". Após ultrapassar esse tempo de vida, o mundo deixa de ser utilizável pelos

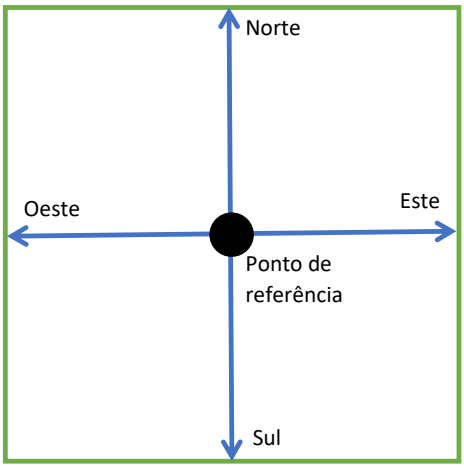


Figura 3.17: Definição do "Mundo do THYME".

utilizadores conectados e deixará de estar disponível para novos utilizadores se poderem conectar, dando por terminado a atividade desse "Mundo do THYME".

A dimensão das células é definida dependo da tecnologia de comunicação utilizada. No caso do Bluetooth e Wi-Fi Direct, a dimensão das células é calculada conforme o alcance da tecnologia. No caso do Wi-Fi, o tamanho das células é definido por um paramento de configuração, sendo 30 metros por predefinição. O tamanho da célula tem de ser sempre inferior ao alcance da tecnologia de forma a permitir a comunicação dos nós entre células.

O espaço geográfico definido pode representar um quadrado ou um retângulo, onde seja possível a divisão exata pelo tamanho das células, caso contrário o mundo será inválido e será assumido um mundo padrão. O mundo padrão, com comunicação via Wi-Fi, é definido por 30 metros em todas as direções (norte, sul, este e oeste) e o tamanho das células é de 30 metros, originando 4 células.

A classe que representa o "Mundo do THYME" e contém estas informações é denominada de *World*. Na Listagem 3.2 podemos ver um exemplo da configuração de um objeto do tipo *World*.

Divisão das células. A divisão do espaço geográfico em células é efetuada pelo módulo "Gestor da Grelha" (Secção 3.2) e depende, obviamente, da configuração inicial do "Mundo do THYME". De acordo com o mundo definido, este vai ser dividido em quadrados de tamanho igual, com o tamanho definido como referido na Secção 3.7, sendo que os limites definidos devem permitir uma divisão exata pelo tamanho de células, de forma a que todo o espaço geográfico definido seja abrangido pelas células. Caso contrário será utilizado um mundo padrão com as seguintes características:

- Centro do mundo - Posição atual do nó;
- Norte - 30 metros;

- Sul - 30 metros;
- Este - 30 metros;
- Oeste - 30 metros;
- Tamanho das células - 30 metros[3];
- Nome - Thyme;
- Duração - 7200000 milissegundos (2 horas).
- Originando assim quatro células, com o centro na posição atual do nó e sendo o mundo disponível por 2 horas.

A identificação das células são representados por inteiros e a numeração é efetuada da esquerda para a direita de baixo para cima como se pode verificar na Figura 3.18.

6	7	8
3	4	5
0	1	2

8	9	10	11
4	5	6	7
0	1	2	3

Figura 3.18: Identificação das células.

3.8 Encaminhamento

Na abordagem do Thyme materializada nesta dissertação decidimos utilizar as características da TDG-C com um protocolo de encaminhamento geográfico, de forma a facilitar o encaminhamento das mensagens. O encaminhamento das mensagens é efetuado através dos nós virtuais utilizando um protocolo de encaminhador geográfico, uma adaptação do protocolo "Greedy perimeter stateless routing" (GPSR) [42]. Este protocolo de encaminhamento toma decisões gananciosas, enviando as mensagens para o nó que consegue alcançar, mais próximo do nó de destino da mensagem em questão.

Originalmente, de acordo com [4], a comunicação é efetuada exclusivamente entre células, não existindo comunicação entre nós individuais. Mas o nosso sistema publicador/subscritor necessita de enviar mensagens para nós individuais em diversas ocasiões – a quando de uma resposta de confirmação de uma operação ou a quando de uma notificação de uma subscrição efetuada por um determinado nó.

De forma a colmatar esta necessidade, no Thyme é possível enviar mensagens direcionadas para uma célula, seta a laranja na Figura 3.19, quando tratamos de uma operação

[3]Utilizando a tecnologia de comunicação Wi-Fi.

como uma publicação, uma subscrição ou outras similares. Contudo, também é possível enviar mensagens para um determinado nó, seta a vermelho na Figura 3.19, quando é necessário enviar uma resposta ou uma notificação para um nó em específico.

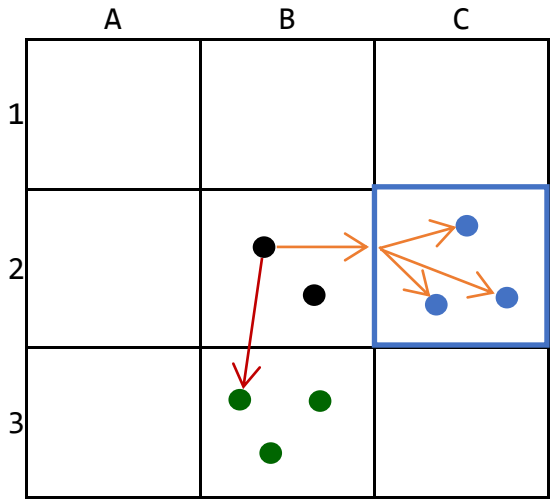


Figura 3.19: Destinos possíveis das mensagens enviadas.

3.9 Movimento

Perante um contexto onde a mobilidade dos nós é uma realidade que temos de ter em conta, achamos necessário que o THYME fosse resistente à mobilidade dos nós sem condicionar o desempenho do sistema.

No THYME os nós podem mover-se espontaneamente dentro da mesma célula ou entre células. Sendo que apenas os nós estáveis, isto é, os nós que não estão em movimento, participam no encaminhamento das mensagens e fazem parte das células da TDG-C.

Quando um nó começa a movimentar-se, envia uma mensagem para os restantes nós avisando que está a movimentar-se e deixa de enviar as mensagens periódicas de "Hello". Desta forma deixa de pertencer à sua célula da TDG-C, não podendo ser escolhido para responder a pedidos efetuados na sua célula, mas poderá continua a receber atualizações do estado da célula, se o movimento for efetuado dentro da célula, pois os dados são enviados para todos os nós dentro da célula.

Após estabilizar, o nó pode continuar na mesma célula ou pode ter-se movido para uma célula diferente e parado nesse local.

Se continuar na mesma célula apenas tem de voltar a enviar as mensagens periódicas de "Hello" e assim voltar a pertencer à célula e respetivamente à TDG-C.

Se o nó, após terminar o seu movimento, parar numa célula diferente da célula de partida, existem uma série de informações que este nó tem de atualizar.

Estado da célula - Como referido na Subsecção 3.6.1, quando um novo nó entra na célula, este tem de ser atualizado com o estado da célula atual para que possa responder aos pedidos de forma uniforme com os restantes elementos da célula. Desta forma o novo nó tem de atualizar-se com o estado da sua nova célula, eliminando os dados que possuía da sua antiga célula.

Réplicas passivas - Os nós que obtiveram os dados por vontade própria são utilizados como réplicas passivas desses mesmo dados. Como o nó mudou de célula, é necessário atualizar a informação em todas as suas réplicas passivas.

Subscrições - As notificações são enviadas para os utilizadores que efetuaram subscrição sobre a *tag* em questão. Se o nó se movimentou para outra célula, as notificações serão envidas para a célula onde o nó efetuou a subscrição. De forma a continuar a receber as notificações das suas subscrições é necessário atualiza-las com a sua posição atual.

Dados publicados - Se o nó que se moveu efetuou publicações no sistema, quando se move para outra célula é necessário atualizar as suas publicações. O nó tem de informar à célula responsável pelas *tags* dos objetos que publicou, indicando que se encontra numa célula diferente da célula onde foi publicado o objeto, passando o nó publicador a ser considerado uma réplica passiva, pois possui o objeto por vontade própria.

Os nós que se moverem para fora do espaço geográfico definido deixarão de fazer parte do sistema e não participarão no mesmo.

3.9.1 Movimento e as notificações

Se uma notificação for enviada quando o nó subscritor se encontra em movimento, essa notificação pode nunca chegar ao destino, pois se o subscritor estiver a mover-se numa célula diferente da célula registada poderá nunca receber a notificação. Este problema pode ser resolvido sendo enviada uma mensagem a quando do início do movimento, para as células responsáveis pelas subscrições de forma a serem guardadas as notificações num *buffer* e após a receção da mensagem de nova localização desse dispositivo, são enviadas as notificações, garantindo assim que estas chegam ao subscritor. Com esta solução seria necessário adicionar um tempo de vida às notificações pois o nó poderia mover-se para fora do sistema, deixando o sistema, não sendo necessário, nem vantajoso, guardar as subscrições por tempo indeterminado.

Outra solução para a resolução deste problema seria a utilização de um mecanismo de "confirmação negativa", onde a camada de Encaminhamento informava o sistema que a mensagem não foi entregue, porque o nó não se encontrava na célula esperada. Esta solução necessitaria também de um *buffer* de notificações, mas permitiria diminuir o

número de mensagens enviadas pois, não seria necessário enviar uma mensagem sempre que um nó se movimentasse.

3.10 Detalhes de Implementação

Existem alguns detalhes na implementação do Thyme que devem ser explicadas de forma a permitir uma melhor compreensão do sistema. Ao longo das próximas subsecções serão explicadas as decisões de implementação tomadas, algumas adaptações devido ao contexto e alguns pormenores do Thyme.

3.10.1 Tempo no Thyme

O tempo é uma dimensão de primeira ordem para o Thyme, como temos vindo a referir ao longo deste documento. Visto ser uma componente tão importante, é fundamental que todos os dispositivos que participem no sistema tenham os seus relógios sincronizados, caso contrário o sistema poderá não funcionar da forma correta.

Esta exigência, para o bom funcionamento do nosso sistema, parece-nos razoável e de fácil cumprimento uma vez que a maioria dos dispositivos móveis possuem os seus relógios sincronizados com o tempo fornecido pela rede móvel.

O incumprimento desta exigência não levará à impossibilidade da utilização do Thyme mas levará certamente a um funcionamento incorreto de acordo com o pretendido pelo utilizador.

3.10.2 Protocol Buffers

A escolha do mecanismo e do formato de serialização dos dados, de forma a permitir o seu envio pela rede, pode ser determinante no comportamento do sistema.

Existem vários mecanismos e formatos de serialização como *XML*, *JSON*, *Java Serialization* ou *Protocol Buffers*. Depois de terem sido analisados alguns artigos relacionados com o assunto, como [43] [52] [81], concluiu-se que o mais indicado para o nosso sistema seria a utilização de *Protocol Buffers*.

O *Protocol Buffers* [63] é um mecanismo de serialização de dados estruturados desenvolvido pela Google, que pode ser utilizado em várias plataformas. A escolha por este método de serialização deve-se a um leque alargado de razões. Em primeiro lugar devido à velocidade de comunicação, isto porque esta solução permite uma velocidade de transmissão bastante mais rápida que os restantes, de 20 a 100 vezes mais rápido que o XML, referido pela autores em [64], mas também comprovado em [81]. Esta característica é fundamental perante o contexto onde nos inserimos.

Além da velocidade de comunicação, também o tamanho das mensagens foi um ponto crucial na decisão deste mecanismo, sendo os *Protocol Buffers* capazes de produzir mensagens mais pequenas comparativamente às restantes opções, foi mais um ponto a favor.

Por fim, o facto do mecanismo de definição dos dados ser bastante rápido e eficiente contribui a favor dos *Protocol Buffers*. Para a definição de uma mensagem apenas é necessário definir num ficheiro *.proto*, de forma numerada, qual o tipo de dados a ser serializados e o nome referente a esses dados de forma a ser identificado. É possível definir uma grande variedade de tipos de dados[4]. De seguida apenas é necessário executar o compilador dos *Protocol Buffers* que dará origem a classes que permitem o acesso e o preenchimento do dados dos mensagem.

A quando da criação da mensagem apenas é necessário efetuar o preenchimento dos dados através dos métodos *"set"*, e no final gerar os bytes que serão enviados, através do método *"build()"*. Na receção apenas é necessário efetuar o acesso e recuperação dos dados através dos métodos *"get"* e criação do objeto que foi enviado.

3.10.3 Instâncias do Thyme

O Thyme é um sistema que pode ser utilizado para os mais diversos casos de uso e por esse motivo pode ser necessário que os utilizadores ou outros serviços tenham a necessidade de usar vários "Mundos do Thyme" dentro do mesmo sistema.

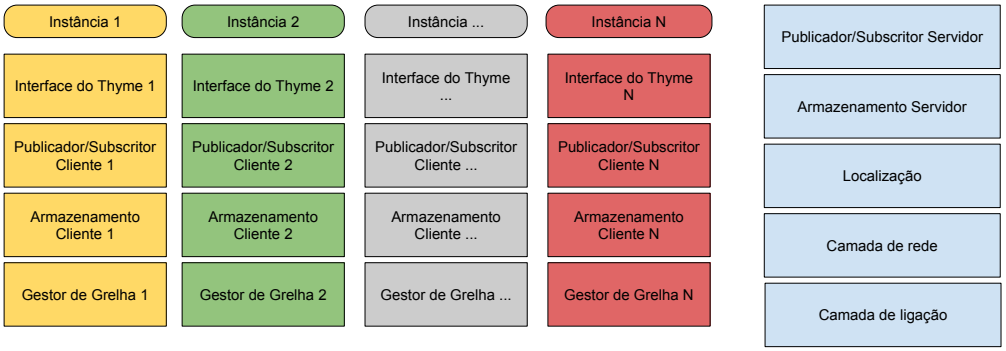


Figura 3.20: Várias instâncias do Thyme.

A pensar nesta situação, o Thyme possibilita a criação de várias instâncias no mesmo sistema — cada instância representa um "Mundo do Thyme" — sendo que cada instância é independente a nível do cliente mas o servidor é partilhado com todas as instâncias. Isto permite que cada instância possa efetuar as suas publicações, as suas subscrições e receber as suas notificações independentemente das restantes instâncias, e garante que a parte de servidor efetue o seu trabalho independentemente da instância em questão.

A Figura 3.20 apresenta os objetos de cada módulo que são criados quando são iniciadas **N** instâncias. A cada nova instância é criado um novo objeto da vertente "Cliente" do *Publicador/Subscritor* e *Armazenamento* mas também um objeto da *Interface do* Thyme e do *Gestor de grelha*. Sendo que a vertente "Servidor" do *Publicador/Subscritor* e *Armazenamento*, assim como a *Localização*, *Camada de rede* e *Camada de ligação*

[4]Os tipos de dados suportados encontram-se em [65].

são independentes da instância, sendo apenas criado um objeto utilizado pelas várias instâncias.

As instâncias do THYME são diferenciadas através do identificador único do "Mundo do THYME" que cada instância representa, identificado por um inteiro no sistema.

3.10.4 Localização

O funcionamento do THYME necessita da localização dos nós, pois é através da localização do nó que se calcula qual a célula onde o nó se encontra, de forma a pertencer ao nó virtual, e também permite proceder ao cálculo da distância entre nós de forma a enviar mensagens para os nós mais próximos quanto possível.

Neste caso, é utilizado o GPS como mecanismo de localização, mas poderiam ser utilizadas outras soluções, como por exemplo os mecanismos apresentadas em trabalhos como [14, 66].

De forma a identificar a localização dos nós utilizamos as coordenadas geográficas Latitude e a Longitude. A coordenada Altitude não foi utilizada por não ser fornecida pelo mecanismo de localização utilizado, o GPS, mas o sistema encontra-se preparado para adicionar esta coordenada se posteriormente for necessário.

A utilização do GPS com mecanismo de localização cumpre os requisitos mínimos exigidos para o sistema mas apresenta alguns desafios para superar, em relação à obtenção de localização, como será abordado na Secção 6.2.

3.10.5 Mecanismo de Movimento

Visto que estamos perante um ambiente real e onde utilizamos dispositivos móveis, é comum existir movimento por parte dos utilizadores do sistema. Por essa razão é necessário a utilização de um mecanismo que detete o movimento do dispositivo.

Na nossa implementação do módulo *Localização* foi utilizado o acelerómetro como sensor de movimento. Detetando a aceleração do dispositivo móvel, é possível detetar se este se encontra em movimento ou em estado de repouso.

A deteção de movimento por parte do sistema acontece quando o acelerómetro deteta que o dispositivo se encontra com uma aceleração superior a um valor base que representa o estado de repouso, configurável no ficheiro de propriedades. Enquanto a aceleração do dispositivo se mantiver superior ao valor base, o sistema considera que o nó se encontra em movimento. Quando o valor da aceleração baixar para valores inferiores ao valor base, o sistema assume que o nó parou e obtém a sua localização três vezes seguidas (valor configurável), de forma obter um posicionamento mais preciso.

O mecanismo de localização utilizado nesta implementação é suficiente mas apresenta certas limitações. O acelerómetro mede a aceleração do dispositivo, que não é o melhor indicador para a deteção do movimento do utilizador. O utilizador pode estar parado mas um movimento brusco do dispositivo pode induzir o sistema em erro, isto porque a

aceleração será superior ao valor base para o estado de repouso. mas o utilizador encontra-se parado.

Como abordado na Secção 6.2, este pode ser um ponto a melhorar num trabalho futuro, implementando outro mecanismo de movimento mais eficiente.

3.10.6 *Time out*

A perda de mensagens é uma realidade que temos de ter em atenção quando estamos perante um contexto onde utilizamos dispositivos móveis e redes de dispositivos móveis. De forma a precaver a perda de mensagens e a garantir a execução das tarefas executadas pelo sistema, sempre que é enviada uma mensagem que necessita de resposta, foi adicionado um tempo de vida a essa mensagem, denominado na gíria de *time out*. Se após esse tempo de vida o sistema não obtiver resposta, essa mensagem é considerada perdida e é reenviada uma nova mensagem com a mesma informação que a anterior.

No caso da operação de obtenção dos dados são efetuadas três tentativas de envio da mensagem, se após as três tentativas o resultado não for obtido, assumimos que não é possível efetuar esta operação e essa informação é transmitida à aplicação para que possa tomar uma ação sobre a situação.

3.10.7 Células vazias

O nosso sistema apresenta alguns detalhes e duas limitação relacionados com as células relacionadas com as células vazias.

Armazenamento. O Thyme garante a persistência dos dados utilizando, maioritariamente, a replicação ativa dos dados (abordada na Secção 3.5), utilizando o facto da TDG-C tratar todos os nós da célula como um nó virtual onde os dados se encontram replicados. Se por algum motivo uma célula, que já esteve populada ficar vazia, não existem garantias da persistência dos dados no sistema, pois se nenhum nó obteve esse objeto por vontade própria, o objeto deixará de estar disponível no sistema pois não existem réplicas disponíveis desse objeto.

Além desse problema, os metadados são armazenados pelo nó virtual, isto é, pela célula responsável pela *tag* dos metadados. Se existir uma célula que deixa de estar populada, esses metadados vão se perder, perdendo-se também subscrições e deixando os subscritores de receber notificações sobre as *tags* que sobrescreveram.

Há formas de ultrapassar essa limitação, como as apresentadas no artigo [4], mas estas ainda não foram incorporadas nesta implementação pelo que a sua utilização está limitada aos cenários em que as células uma vez populadas, não poderão ficar vazias.

Encaminhamento. O encaminhamento do Thyme é, como já foi referido, uma adaptação do protocolo "Greedy perimeter stateless routing" (GPSR) [42] ao contexto que nos encontramos.

O módulo responsável pelo encaminhamento, apresentado na Secção 3.2, está preparado para lidar com a questão das células vazias quando é necessária a comunicação em vários saltos (*multi-hop*). O algoritmo de encaminhamento toma decisões gananciosas, isto é, envia a mensagem para a célula mais próxima do nó/célula de destino mas quando não existe a possibilidade de continuar com o encaminhamento ganancioso, pois existem células vazias, o algoritmo utiliza um encaminhamento com base no perímetro da célula de destino.

Apesar do módulo estar preparado para esse problema, no contexto que nos encontramos e com o tipo de rede que utilizamos, fornecido pela *Camada de ligação* e *Camada de rede*, não é necessário utilizar uma comunicação em vários saltos pois todos os nós se conseguem alcançar, logo é apenas necessário um salto. Devido a esta característica, o encaminhamento não tem problemas com a existência de células vazias pois não interfere no envio das mensagens.

Função de distribuição A tabela de dispersão geográfica baseada em células que é utilizada no THYME indica qual a célula responsável pela respetiva *tag* efetuando uma função de distribuição, sendo o resultado uma das células do "Mundo do THYME" definido. Neste caso, é possível que seja escolhida uma célula que não esteja populada e nesse caso as operação relacionadas com essa *tag* irão falhar, uma vez que estando a célula vazia nenhum nó irá tratar desse pedido.

Uma das formas de resolver esta limitação, abordada nos artigos [4] e [68], seria aproveitar as características do protocolo de encaminhamento que utilizamos, o GPSR [42], para encaminhar as mensagens da célula vazia para uma célula populada sua vizinha. Quando uma mensagem fosse enviada para uma célula vazia, essa mensagem seria encaminhada para uma das suas células vizinhas que estivesse populada, sendo que essa célula seria temporariamente responsável por essa *tag*, armazenando todos dados e respondendo a todas as operações relativas a essa *tag*. Desta forma o utilizador poderia efetuar operações sobre essa *tag* sem problemas. Se a célula originalmente responsável pela *tag* ficasse populada, seria necessário mover todos os dados relativos à sua *tag* da célula temporariamente responsável para a célula originalmente responsável, sendo que a partir desse momento a célula originalmente responsável iria responder aos pedidos sobre a(s) sua(s) *tag(s)*.

Sendo que esta solução não se encontra implementada no nosso sistema, está é uma limitação. Desta forma, se existirem células vazia no sistema, não é possível garantir funcionalidade para todas as *tags*. Pretende-se implementar o suporte apresentado num futuro próximo, de forma a mitigar esta limitação, sendo um dos trabalhos futuros de maior relevância.

3.10.8 Endereço.

Cada nó no Thyme contém um endereço constituído por um *UUID* [45] e pelo identificador da sua célula, um inteiro. Em 3.1 podemos observar como são constituídos os endereços dos nós e em 3.2 está representado um exemplo que poderia ser um endereço de um nó no nosso sistema.

$$(UUID : idCell) \tag{3.1}$$

$$(123e4567 - e89b - 12d3 - a456 - 426655440000 : 0) \tag{3.2}$$

Este endereço é obtido através da *Camada de rede* e é utilizado para permitir a comunicação entre os vários nós do sistema. O *UUID* do endereço é gerado aleatoriamente quando o Thyme é iniciado e não é alterado durante a sua execução, já a parte do identificador da célula é alterado sempre que o nó se movimentar e estabilizar numa célula diferente da célula inicial.

3.10.9 Tecnologia de comunicação

A comunicação entre os vários dispositivos do sistema é efetuada através de uma rede de dispositivos móveis que é formada utilizando as tecnologias inerentes aos dispositivos móveis. Como referido na Secção 3.2, esta rede pode ser formada utilizando o Wi-Fi, o Bluetooth ou o Wi-Fi Direct. A escolha de qual a tecnologia de comunicação a ser utilizada é uma informação retirada do contexto do dispositivo móvel. Isto quer dizer que será utilizada a tecnologia de comunicação que estiver ativa no dispositivo móvel. Se o utilizador tiver o Wi-Fi ligado será essa tecnologia utilizada para efetuar a comunicação. Sendo verdade também para as restantes tecnologias.

Porém, o utilizador pode possuir as várias tecnologias ativas em simultâneo no seu dispositivo, neste caso o Thyme opta pela seguinte ordem: 1) Wi-Fi 2) Bluetooth 3) Wi-Fi Direct . Esta ordem foi escolhida tendo em conta a taxa de utilização de cada tecnologia, isto é, é mais comum a utilização do Wi-Fi em detrimento do Bluetooth e por fim o Wi-Fi Direct.

É importante ter em atenção que apenas os utilizadores que utilizam a mesma tecnologia de comunicação conseguem estabelecer conexão entre si.

3.10.10 Lidar com *Churn*

Devido às características dos dispositivos para os quais estamos a trabalhar, dispositivos móveis, é importante ter em atenção os vários cenários possíveis tendo em conta as suas características. Por essa razão e devido à mobilidade que estes dispositivos possuem e aos recursos energéticos limitados, os utilizadores do nosso sistema podem sair por livre e espontânea vontade do sistema, ou da área definida pelo "Mundo do Thyme" ou podem

ter sido obrigados a sair do sistema por limitações energéticas ou por qualquer outra falha do dispositivos. Podendo outros entrar a qualquer momento.

Tendo em conta as características do nosso sistema, o *churn* – entrada e saída de dispositivos no sistema – é precavido com a replicação dos dados de forma a evitar que a saída de qualquer elemento do sistema possa causar a perda de dados no sistema. Além disso, quando um nó sai definitivamente da rede e possui subscrições no sistema, foi decidido eliminar essas subscrições, caso contrário poderiam ser enviadas mensagens de notificação dos objetos subscritos desnecessariamente pois o nó não se encontra no sistema.

A eliminação das subscrições dos nós que abandonaram a rede tem de ser bem coordenada de forma a não eliminar subscrições de nós que se encontram em movimento, podendo ser confundidos com saídas da rede. Dessa forma, utilizamos as mensagens periódicas de "Hello" para controlar a saída dos dispositivos da rede. Se o dispositivo deixar de enviar mensagens de "Hello" durante um período configurável através do ficheiro de propriedades, definido como padrão de 10 minutos, será assumido que o dispositivo deixou o sistema e dessa forma serão eliminadas as suas subscrições.

3.11 Sumário

Neste capítulo foi apresentado o modelo Thyme, assim como a sua implementação para redes de dispositivos móveis Android. Foram apresentadas as características do sistema, o seu funcionamento, assim como as decisões que foram tomadas para que fosse possível efetuar a implementação do modelo Thyme no contexto pretendido para este trabalho. Após este capítulo é possível compreender como opera o Thyme e como este funciona na sua implementação para redes de dispositivos móveis.

O próximo capítulo será dedicado à aplicação que foi desenvolvida como caso de estudo para a nossa implementação do Thyme, a "Galeria de Fotos Thyme". Ao longo do capítulo será possível compreender, as suas características e as funcionalidades da aplicação, e de que maneira o utilizador pode usufruir dessas funcionalidades.

Capítulo 4

Caso de estudo: Galeria de Fotos Thyme

A implementação do Thyme em Android pode ser utilizada nos mais diversos casos de uso, com referimos no Capítulo 1. Neste capítulo é apresentado o caso de estudo - "Galeria de Fotos Thyme". O capítulo inicia-se com a Secção 4.1 que apresenta uma visão geral da aplicação, explicando a sua finalidade e o seu funcionamento geral. Em seguida, na Secção 4.2 é demonstrado como se inicia e configura a aplicação e qual a razão de tal configuração. Nas restantes secções, da Secção 4.3 à Secção 4.4.6, são apresentadas as operações disponíveis pela aplicação, onde são demonstradas e explicadas cada uma das funcionalidades disponíveis.

4.1 Visão geral

A "Galeria de Fotos Thyme" é uma aplicação para dispositivos móveis com o sistema operativo Android que permite a partilha de fotografias entre dispositivos conectados através de uma rede sem fios.

Esta aplicação utiliza o sistema Thyme para permitir a partilha e disseminação de fotografias e para garantir a persistência dos dados partilhados.

A nossa aplicação permite aos utilizadores:

- publicarem as suas fotografias no sistema indicando quais as *tags* relacionadas com o objeto;
- subscreverem *tags* pelo qual estão interessados, de forma a receber notificações sobre os objetos publicados com essa *tag*;
- após receberem notificações de uma fotografia os utilizadores podem obter esse objeto ou podem colocar esta fotografia disponível para descarregar mais tarde;

- é possível ainda remover subscrições efetuadas pelos próprios utilizadores; e
- possibilidade de *despublicar* objetos publicados por si no sistema, tornando-os inacessíveis para novos subscritores.

A "Galeria de Fotos Thyme" permite que os utilizadores criem galerias partilhadas – como se de uma pasta partilhada se tratasse – sendo que outros utilizadores se podem conectar às mesmas. Apenas os utilizadores conectados à mesma galeria podem partilhar fotografias entre si.

O objetivo do desenvolvimento desta aplicação é testar e comprovar o funcionamento do sistema desenvolvido nos mais diversos casos de uso à qual nos propusemos, como festas de aniversário, reuniões, eventos sociais, eventos esporádicos em geral, entre outros, como referimos na Secção 1.2.

Esta aplicação pode ser instalada e executada em qualquer dispositivo com a versão 5.0 (*Lollipop*) ou superior do sistema operativo Android, sem a necessidade de fazer qualquer alteração ao sistema ou permissões de administrador.

Para garantir um bom funcionamento da aplicação é fundamental que os relógios dos dispositivos estejam sincronizados - através da rede móvel, por exemplo, uma vez que o tempo é uma dimensão de primeira ordem para o sistema utilizado, como abordado na Secção 3.10.1.

Devido à utilização do Thyme, esta aplicação não necessita de acesso à Internet e a comunicação entre dispositivos móveis é estabelecida utilizando as tecnologias de comunicação inerentes a estes dispositivos, como o Wi-Fi, Bluetooth ou Wi-Fi Direct.

4.2 Inicialização da "Galeria de Fotos Thyme"

Quando um utilizador inicia a aplicação, é necessário aguardar que o sistema efetue uma procura na rede, representado na Figura 4.1, que é utilizada para detetar se existem outros dispositivos na mesma rede, verificando a existência de galerias às quais o novo utilizador se pode conectar. Após esta procura na rede podem existir duas situações possíveis:

1) Não é encontrada nenhuma galeria disponível, por isso é necessário configurar uma nova galeria (Secção 4.2.1); ou
2) Existem galerias disponíveis, sendo que o utilizador pode selecionar uma galeria para se conectar (Secção 4.2.2) ou pode configurar uma nova galeria (Secção 4.2.1) se assim desejar.

4.2.1 Configuração de uma Nova Galeria

Se após a pesquisa na rede que foi efetuado na inicialização da aplicação não forem encontradas galerias disponíveis, surgirá ao utilizador um cenário semelhante à Figura 4.2a.

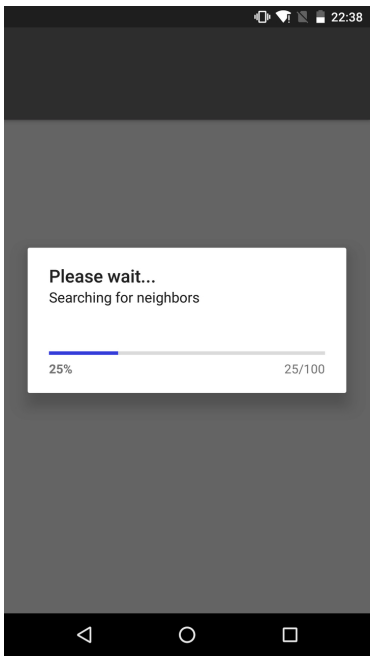


Figura 4.1: Ecrã representativo da procura de galerias disponíveis.

Nesta situação é possível escolher uma de duas opções: 1) Criar uma nova galeria, através da opção New Gallery ; ou 2) Voltar a pesquisar por galerias potencialmente existentes, clicando em Find.

Optando o utilizador por criar uma nova galeria, é necessário configurar a nova galeria, introduzindo:

Name - o nome da galeria, de forma a ser identificável por outros utilizadores;

North - a distância a norte abrangida pela galeria (em metros);

South - a distância a sul abrangida pela galeria (em metros);

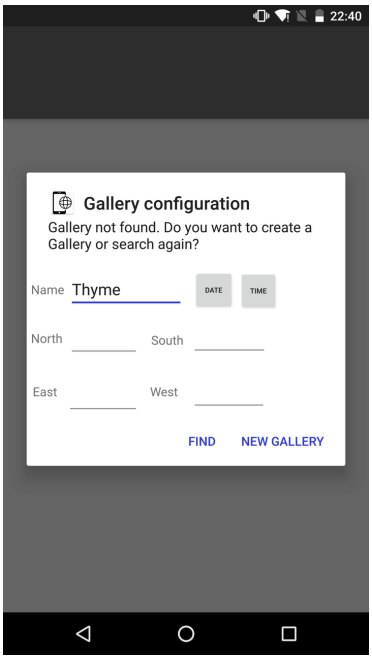
East - a distância a este abrangida pela galeria (em metros);

West - a distância a oeste abrangida pela galeria (em metros);

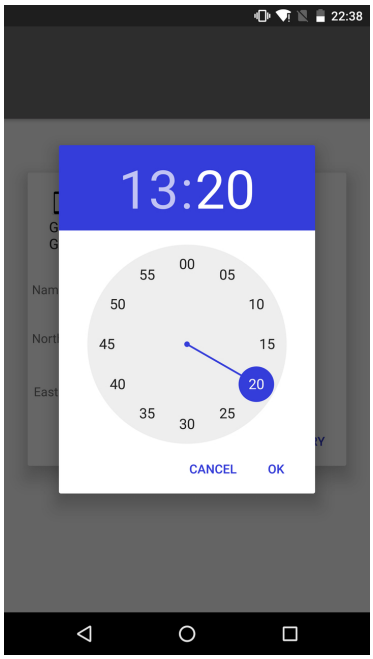
Time - o hora de termino da galeria (em horas e minutos) (Figura 4.2b) ;

Date - a data de termino da galeria (dia, mês e ano).

Relacionando com o Thyme, esta configuração é utilizada para criar um novo "Mundo do Thyme", abordado na Secção 3.7, que definirá onde esta galeria vai ser ativa e o tempo de vida em que é possível a utilização desta galeria. Após esta configuração, a aplicação está pronta a ser utilizada pelo utilizador e a galeria encontra-se disponível para outros utilizadores se conectarem (Subsecção 4.2.2) e efetuarem as operações desejadas.



a Ecrã de nova galeria.



b Introdução da duração da galeria.

Figura 4.2: Configuração de uma nova galeria.

4.2.2 Galeria Existente

Após a pesquisa na rede e se existirem galerias ativas nas proximidades, o nome dessas galerias serão apresentadas ao utilizador como no exemplo da Figura 4.3.

Observando as galerias existentes, o utilizador pode optar por conectar-se a uma das galerias existentes, selecionando o nome da galeria pretendida e selecionando a opção Select ou pode optar por criar uma nova galeria, clicando em New Gallery e seguindo os passos apresentados na Subsecção 4.2.1.

4.2.2.1 Várias galerias

Após ser configurada uma nova galeria (Subsecção 4.2.1) ou depois da conexão a uma galeria já existente (Subsecção 4.2.2), a aplicação encontra-se funcional e permite a partilha e disseminação de fotografias entre os utilizadores conectados à mesma galeria. No entanto, os utilizadores podem desejar participar em várias galerias de forma a partilhar e receber fotografias de todas essas galerias. Neste caso, é utilizada a característica das múltiplas instâncias do Thyme, apresentadas na Subsecção 3.10.3, sendo possível conectar-se a várias galerias no mesmo processo da aplicação.

A conexão a uma nova galeria ou a troca entre galerias onde o utilizador participa, é possível através da opção Change Gallery, visível na Figura 4.4. Após a seleção desta opção, a aplicação irá mostrar quais as galerias disponíveis para conexão e também a opção de criar uma nova galeria – tal como na Figura 4.3.

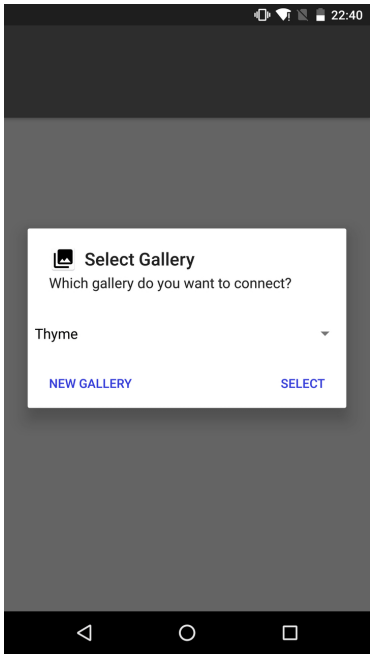


Figura 4.3: Ecrã de seleção de galerias existentes.

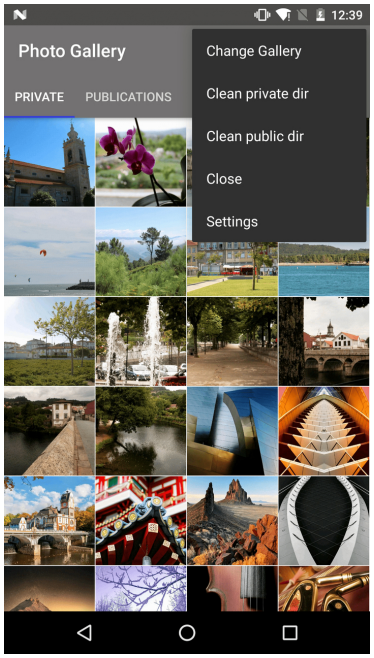


Figura 4.4: Mudança entre galerias.

4.3 Ecrã inicial

Após terminar a configuração inicial da galeria do Thyme, surge o ecrã principal da nossa aplicação, representada na Figura 4.5.

Neste ecrã inicial é possível observar quatro separadores que apresentam fotografias nos diferentes estados relativos ao utilizador.

Private - Este separador apresenta as fotografias privadas do utilizador, isto é, todas as fotografias presentes na galeria do dispositivo móvel. Estas fotografias podem ser publicadas no sistema;

Publications - Este separador exibe todas as fotografias que foram publicadas pelo utilizador. Através deste separador é possível *despublicar* as fotografias publicadas;

Downloads - Todas as fotografias obtidas pelo utilizador são apresentadas neste separador;

Available - Neste separador são exibidas todas as fotografias que foram notificadas ao utilizador mas o utilizador decidiu torná-las disponíveis para obtenção posteriormente.

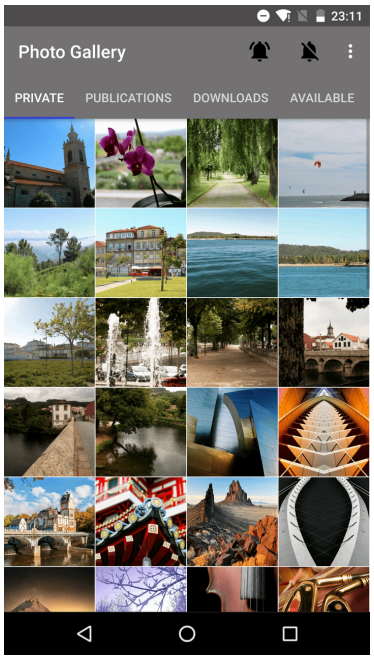


Figura 4.5: Ecrã principal.

Este ecrã inicial é o ponto de partida para a maioria das funcionalidades da nossa aplicação, como a publicação, subscrição, *dessubscrição* e despublicação. Em seguida vamos explicar como é possível efetuar todas as operações disponíveis para o utilizador.

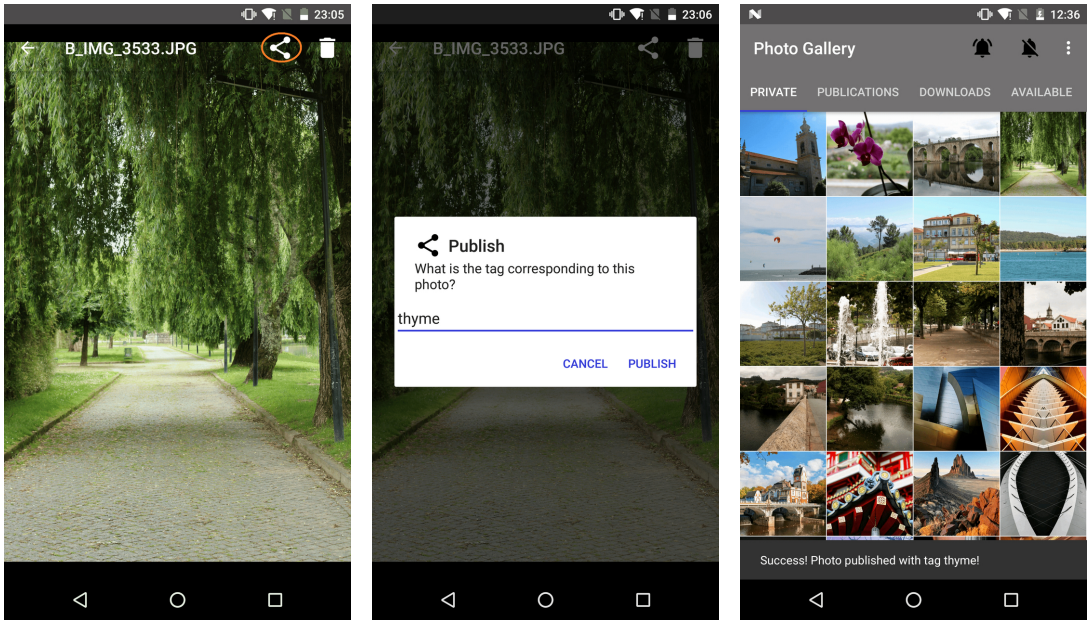
4.4 Operações

A aplicação "Galeria de Fotos Thyme" permite efetuar publicação de fotos privadas, *despublicar* as fotografias publicadas, subscrever *tags*, *dessubscrever tags* previamente subscritas, ser notificado das *tags* subscritas e ainda obter as fotografias notificadas. Nas secções seguintes explica-se como efetuar cada uma dessas operações.

4.4.1 Publicação

A publicação de uma fotografia é apenas possível no separador PRIVATE. Depois de selecionar a fotografia que se pretende publicar, esta surgirá em grande plano – como na Figura 4.6a. Para efetuar a publicação desta fotografia, após selecionar o botão Share – círculo a laranja na Figura 4.6a – é necessário introduzir qual a *tag* ou conjunto de *tags* associadas a esta fotografia e selecionar a opção Publish (Figura 4.6b). No final da operação, o utilizador será informado do sucesso ou falha da publicação. Na Figura 4.6c encontra-se um exemplo em que o utilizador foi informado do sucesso da publicação.

Após serem publicadas com sucesso, as fotografias publicadas deixam de estar visíveis no separador PRIVATE e passaram a ser visíveis no separador PUBLICATIONS.



a Opção para publicação. b Ecrã de publicação. c Sucesso da publicação.

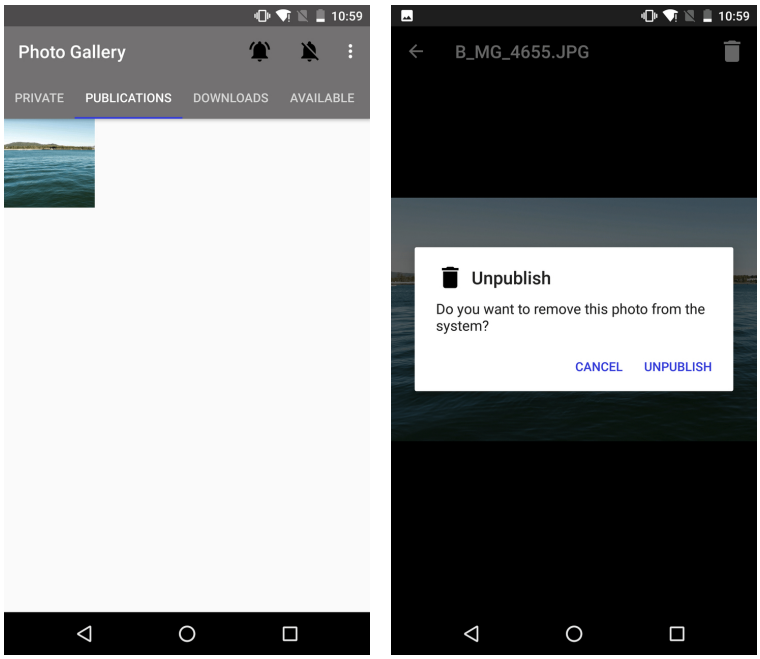
Figura 4.6: Publicação de fotografias.

4.4.2 Despublicação

As fotografias publicadas no sistema podem ser *despublicadas* se o utilizador assim desejar. Essa operação pode ser realizada no separador PUBLICATIONS (Figura 4.7a) e após selecionar a fotografia a *despublicar* surgirá uma caixa de diálogo onde o utilizador terá de confirmar o seu desejo de *despublicar* essa fotografia, como se pode observar na Figura 4.7b. No final da operação o utilizador será informado do estado final da operação.

4.4.3 Subscrição

A subscrição de uma *tag* pode ser efetuada através do botão representado por uma campainha ativa, realçado com um círculo laranja na Figura 4.8a. Este botão está disponível



a Ecrã com fotografias publicadas. b Ecrã de *despublicação*.

Figura 4.7: *Despublicação* de fotografias.

em todos os separadores da aplicação.

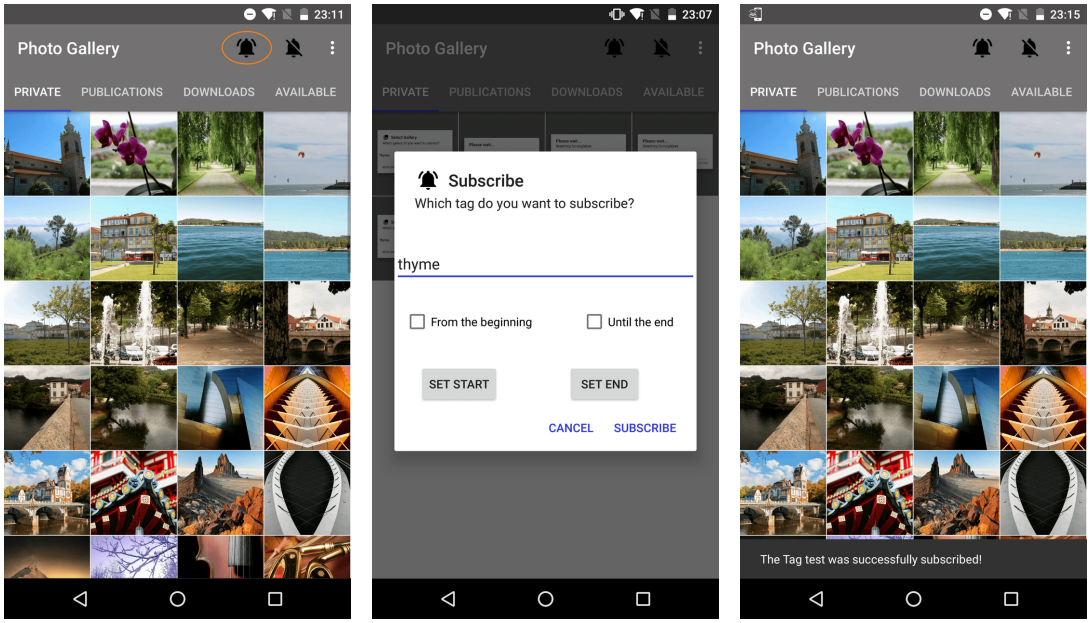
A subscrição necessita que o utilizador indique qual a *tag* que pretende subscrever e qual o tempo de vida da sua subscrição. O utilizador pode optar por subscrever desde o início da galeria – selecionando From the beginning, até ao fim da galeria – com a opção Until the end ou pode selecionar manualmente o início (Set start) e o fim (Set end) do tempo de vida da sua subscrição, utilizando um relógio semelhante ao da Figura 4.2b. Se não for selecionada nenhuma opção, nem introduzido manualmente o tempo de vida, será assumido um tempo de vida com início e fim no momento da subscrição. Após configurada a subscrição é necessário submete-la no sistema através da opção SUBSCRIBE, sendo o utilizador informado do sucesso, como na Figura 4.8c ou falha da subscrição.

A caixa de diálogo da subscrição está representada na Figura 4.8b .

4.4.4 *Dessubscrição*

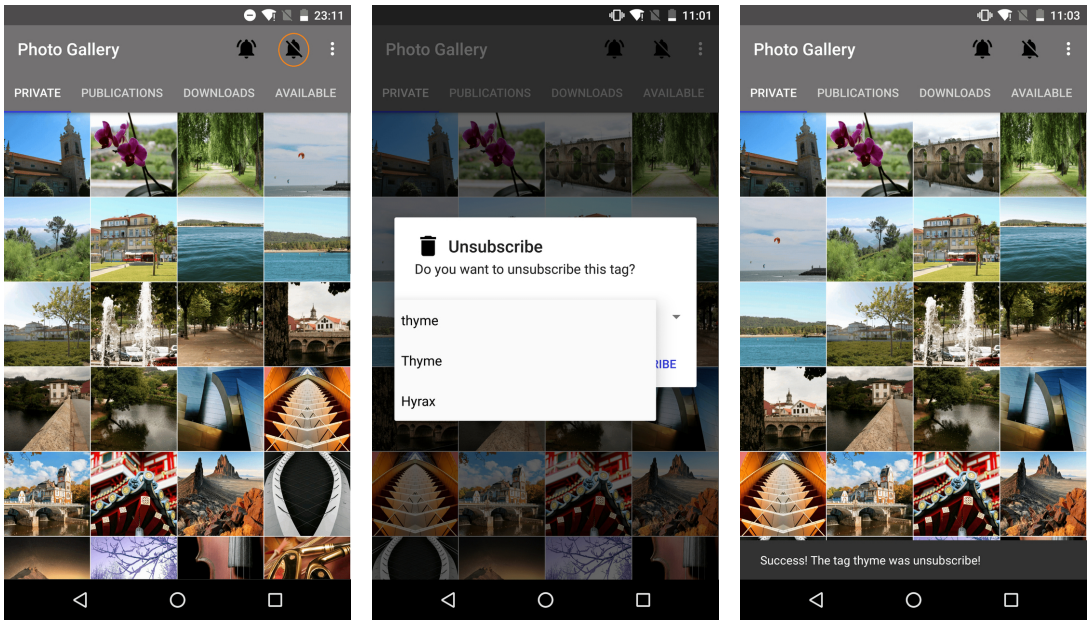
Todas as subscrições efetuadas podem ser anuladas através do botão *dessubscrever* representado na interface com uma campainha silenciada, circulo a laranja na Figura 4.9a.

Após surgir a caixa de diálogo da *dessubscrição*, o utilizador deve selecionar qual a *tag* a *dessubscrever* e submeter a operação selecionando a opção Unsubscribe (Figura 4.9b). O estado final da operação surgirá no ecrã para informar o utilizador, exemplo na Figura 4.9c



a Opção para subscrição. b Ecrã de subscrição. c Sucesso de subscrição.

Figura 4.8: Subscrição de *tags*.



a Opção para *dessubscrição*. b Ecrã de *dessubscrição*. c Sucesso da *dessubscrição*.

Figura 4.9: *Dessubscrição* de *tags*.

4.4.5 Notificação

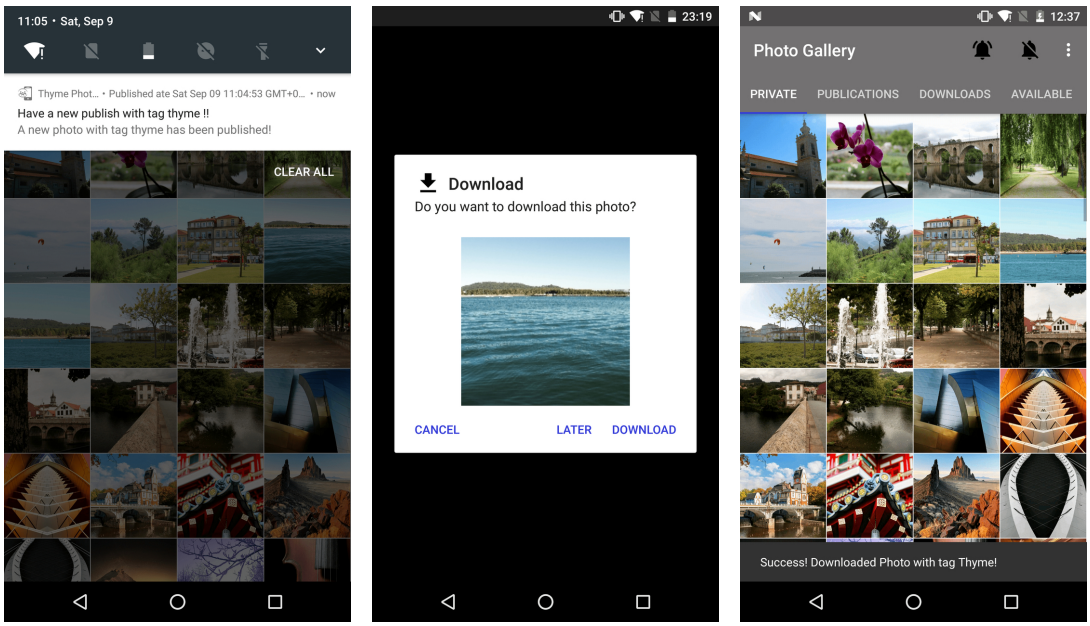
Na "Galeria de Fotos Thyme" apenas é possível obter as fotografias após o utilizador ser notificado da sua existência no sistema (característica do Thyme). Na Figura 4.10a é possível verificar uma notificação da nossa aplicação. A notificação transmite qual é a *tag* relacionada com a fotografia notificada e qual o horário da sua publicação no sistema.

Após selecionar uma notificação surgirá ao utilizador uma caixa de diálogo semelhante à Figura 4.10b. Esta caixa de diálogo apresenta uma miniatura da fotografia notificada e permite efetuar uma de três operações:

Download - Obter esta fotografia (Subsecção 4.4.6);

Later - Tornar esta fotografia disponível para obtenção mais tarde (Subsecção 4.4.7);

Cancel - Ignorar esta fotografia, não sendo possível obtê-la posteriormente.

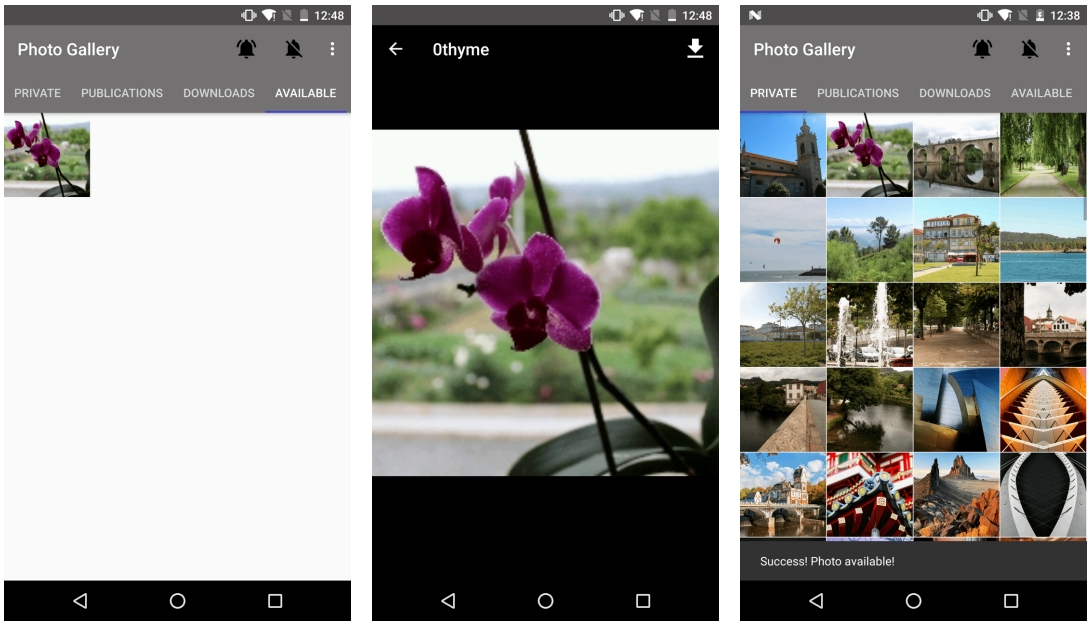


a Notificação. b Ecrã de obtenção. c Sucesso da obtenção.

Figura 4.10: Obter fotografia.

4.4.6 Descarregar

A obtenção das fotografias notificadas ao utilizador é possível, através da opção Download representada na Figura 4.10b. Após esta operação ser efetuada com sucesso (Figura 4.10c), a fotografia será apresentada no separador Downloads e na galeria padrão do dispositivo pois o utilizador contém uma cópia permanente da fotografia no seu dispositivo.



a Fotografias disponíveis para obtenção posterior.

b Obtenção posteriormente.

c Sucesso da obtenção posterior.

Figura 4.11: Obtenção posterior de fotografias.

4.4.7 Descarregar mais tarde

Após receber uma notificação, o utilizador pode optar por não efetuar a obtenção da fotografia de imediato. Dessa forma, a opção Later representada na Figura 4.10b permite deixar essa fotografia disponível para uma obtenção posterior. Quando é selecionada esta opção, a miniatura da fotografia irá ser apresentada no separador Available (Figura 4.11a), sendo que o utilizador poderá selecionar a fotografia posteriormente e efetuar a sua obtenção para o seu dispositivo móvel, clicando no botão "Obter" localizado no canto superior direito da Figura 4.11b.

Ao deslizar a notificação para fora do ecrã, na gíria *"swipe"*, também torna a fotografia disponível para obtenção posterior. Em caso de sucesso, a fotografia ficará disposnivel para obtenção posterior e surgirá uma informação para o utilizador semelhante ao da Figura 4.11c.

4.5 Sumário

No decorrer deste capítulo foi apresentada a aplicação que foi desenvolvida como caso de estudo para implementação do Thyme em Android – "Galeria de Fotos Thyme". Esta aplicação permite aos seus utilizadores publicarem fotografias, associando-lhes *tags*, *despublicar* fotografias previamente publicadas, subscrever *tags*, *dessubscrever tags* previamente subscritas e obter fotografias publicadas no sistema. Este capítulo expôs o funcionamento da aplicação, as suas características e como os utilizadores poderiam efetuar as

operações oferecidas.

No capítulo seguinte, Capítulo 5, será apresentada a avaliação efetuada ao sistema implementado, assim como à aplicação desenvolvida e apresentada neste capítulo. Como seria de esperar, serão também apresentados e discutidos os resultados experimentais dessa avaliação.

Capítulo 5

Avaliação

Neste capítulo são apresentados os resultados da avaliação experimental que foi efetuada para avaliar o comportamento, as características e o funcionamento da implementação do Thyme em dispositivos móveis, assim como da aplicação que desenvolvemos como caso de estudo do sistema. Nesta avaliação foram efetuados testes em ambiente real, com recurso a dispositivos móveis, e em ambiente simulado, utilizando uma implementação simulada do sistema desenvolvido. Na Secção 5.1 são apresentadas as metodologias utilizadas para a execução da avaliação e obtenção dos resultados. Nas secções 5.2 e 5.3, que correspondem respetivamente ao ambiente real e ao ambiente simulado, são apresentados como foram efetuadas as experiências de avaliação e os resultados obtidos com a execução das experiências referidas (Subsecção 5.2.1 e Subsecção 5.3.1). Ao longo da apresentação dos resultados são retiradas conclusões sobre esses valores. Por fim, este capítulo termina com a Secção 5.4 onde é apresentado um sumário da avaliação efetuada.

5.1 Metodologias de avaliação

As experiências que foram efetuadas para avaliar o comportamento, as características e funcionamento do Thyme em redes de diapositivos móveis, foram realizadas em dois ambientes de avaliação diferentes: ambiente real e ambiente simulado.

Em ambiente real foram utilizados dispositivos móveis reais que permitiram avaliar o comportamento do sistema e da aplicação desenvolvida, onde foram realizadas experiências com uma rede de até 32 dispositivos. Nestes testes foi possível avaliar:

- o funcionamento das operações disponíveis pelo Thyme, em ambiente real e com um número aceitável de dispositivos na rede;

- a latência dessas operações;
- o consumo energético do sistema;

A avaliação em ambiente real foi bastante útil para comprovar, na prática, que o sistema implementado e a aplicação desenvolvida sobre este, conseguem dar uma boa resposta a uma utilização em cenários reais.

Em ambiente simulado, foi desenvolvida uma camada de rede para ambientes com fios que permite a comunicação entre os vários nós que eram simulados, permitindo assim a existência de um número bastante elevado de nós. Foi necessário efetuar pequenas adaptações e simular alguns componentes, como por exemplo a localização, pois em ambiente real os dispositivos móveis utilizavam o GPS para obter a localização mas em ambiente simulado foi necessário simular a sua localização. Este ambiente simulado foi desenvolvido em Java e executado apenas numa única máquina.

Nas experiências efetuadas em ambiente simulado o objetivo era avaliar:

- a escalabilidade da implementação do Thyme em larga escala, utilizando elevados números de dispositivos;
- a carga em cada um dos nós, isto é, o número de mensagens recebidas, enviadas e processadas por cada nó no sistema.
- o comportamento do sistema à mobilidade dos nós,
- assim como a entrada e saída de dispositivos (*churn*);

5.2 Ambiente Real

A avaliação efetuada em ambiente real consistiu na execução de uma bateria de experiências onde foram utilizados 2, 4, 6, 8, 12, 16 e 32 dispositivos reais. Estas experiências foram direcionadas para a avaliação do funcionamento das operações disponibilizadas pelo sistema e fornecidas pela aplicação, obter os valores de latência e o consumo de energia.

Além disso, foram efetuados testes para comparar a latência das operações utilizando diferentes tecnologias de comunicação. Neste caso efetuamos experiências de forma a comparar a comunicação utilizando um ponto de acesso Wi-Fi e utilizando Bluetooth. Nestas experiências apenas foi possível avaliar redes com 2, 4 e 6 dispositivos móveis.

Nos testes realizados com 2, 4, 6, 8, 12 e 16 dispositivos foram utilizados exclusivamente dispositivos Nexus 9 e no conjunto de testes realizados com 32 dispositivos, foram utilizados todos os tipos de dispositivos cuja marca, modelo e características estão apresentadas na Tabela 5.1. Na Figura 5.1 é possível observar o conjunto de dispositivos utilizados nestas experiências.

O conjunto de experiências realizadas consistiu em cenários experimentais bem definidos de forma a simular uma utilização real. Esses cenários encontram-se representado

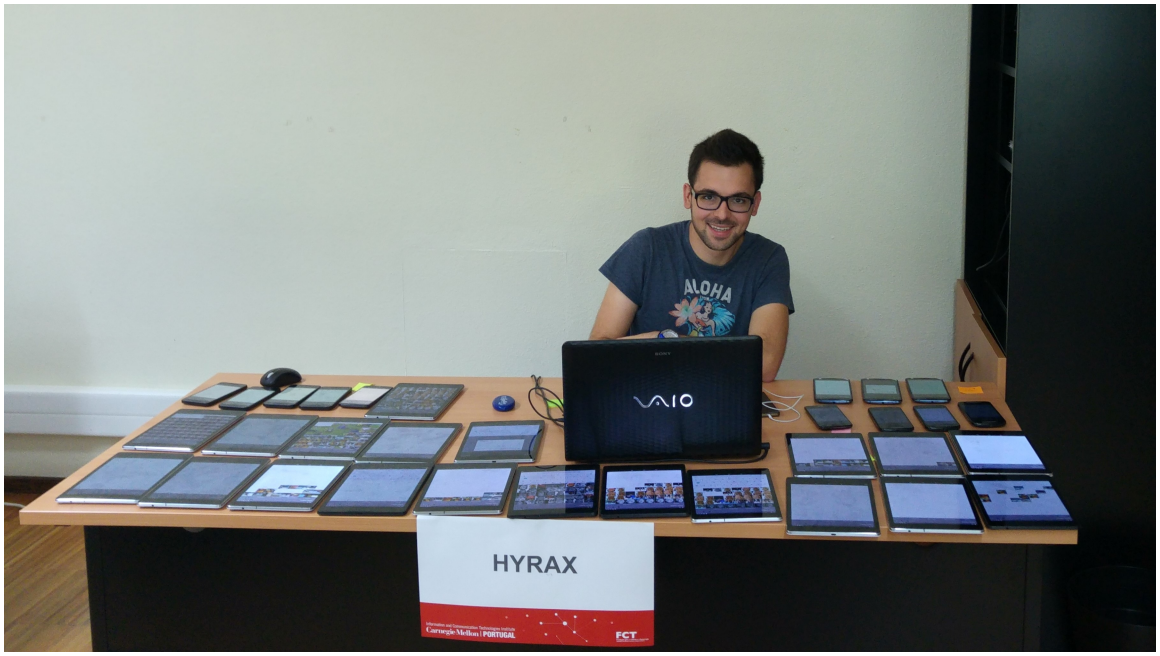


Figura 5.1: Dispositivos utilizados em ambiente real.

Tabela 5.1: Dispositivos utilizados em ambiente real.

Dispositivo	**HTC Nexus 9**	**Motorola Nexus 6**	**LG Nexus 5X**	**Motorola Moto G (2nd gen)**
CPU	Dual-core 2.3 GHz Denver	Quad-core 2.7 GHz Krait 450	Hexa-core (4x1.4 GHz Cortex-A53 & 2x1.8 GHz Cortex-A57)	Quad-core 1.2 GHz Cortex-A7
RAM	2 GB	3 GB	2 GB	1 GB
Armazenamento	16 GB	32 GB	16 GB	8 GB
Bateria	Li-Po 6700 mAh	Li-Po 3220 mAh	Li-Po 2700 mAh	Li-Ion 2070 mAh
SO	Android 7.1.1 (Nougat)	Android 7.1.1 (Nougat)	Android 7.1.1 (Nougat)	Android 7.1.1 (Nougat)
Wi-Fi	Wi-Fi 802.11 a/b/g/n/ac	Wi-Fi 802.11 a/b/g/n/ac	Wi-Fi 802.11a/b/g/n/ac	Wi-Fi 802.11b/g/n
Bluetooth	4.1	4.1	4.2	4.0

ao longo de uma linha de tempo nas Figuras 5.2, 5.3, 5.4 e 5.5 onde estão representadas o cenário 1, cenário 2, cenário 3 e cenário 4 respetivamente. Ao longo dos cenários são utilizadas todas as funcionalidades fornecidas pelo THYME e disponibilizadas pela aplicação: os nós publicam fotografias, subscrevem *tags* no passado e no futuro, recebem notificações, obtêm os dados dos quais foram notificados, *despublicam* fotografias e *desubscrevem tags*.Nas experiências efetuadas foi utilizado um "Mundo do THYME" com duas células, sendo que os nós estavam distribuídos, em número igual pelas duas células. A escolha dos nós que efetuam as operações é efetuada de forma aleatória, tendo em conta as restrições colocadas pelos cenários.

É importante notar que as operações eram efetuadas consecutivamente, apenas com um intervalo de 5 segundos entre operação. Em todas as operações em que a *tag* era relevante – publicação, subscrição e *dessubscrição* – foi utilizada a mesma *tag*: *thyme*. Os comandos de execução das operações foram efetuados de forma automática, utilizando *scripts* desenvolvidos em *Python*, onde era utilizado o *Android Debug Bridge* (ADB), via Wi-Fi, para comunicar com os dispositivos móveis.

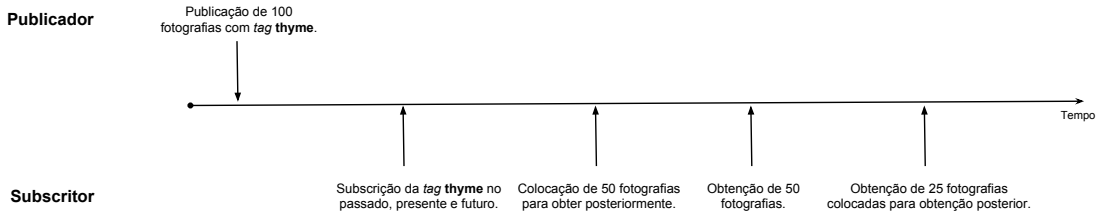


Figura 5.2: Cenário 1.

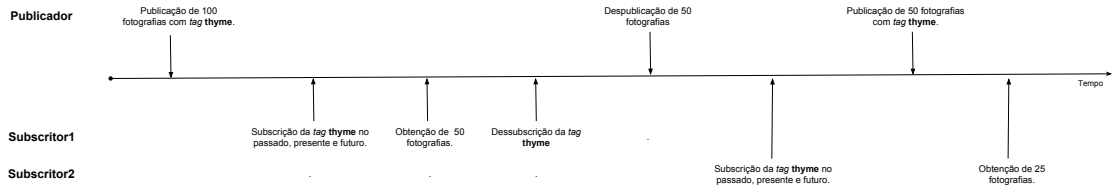


Figura 5.3: Cenário 2.

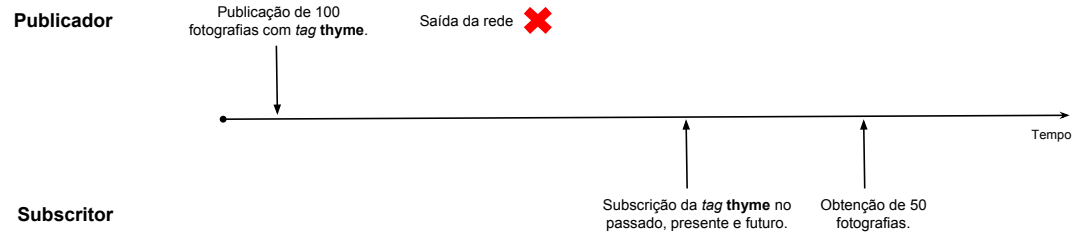


Figura 5.4: Cenário 3.

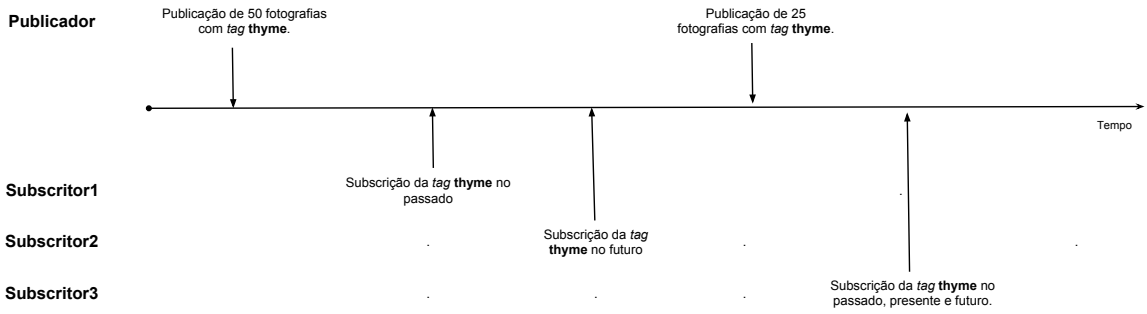


Figura 5.5: Cenário 4.

Todos os testes realizados, exceto os utilizados para a comparação entre tecnologias, foram efetuados com os dispositivos móveis conectados a um ponto de acesso Wi-Fi.

Nos gráficos apresentados nesta secção, é importante referir que a barra de erro corresponde ao desvio padrão dos resultados.

5.2.1 Resultados experimentais

Nesta secção são apresentados os resultados experimentais obtidos segundo os tópicos: funcionalidade das operações, latência das operações e o consumos de energia.

5.2.1.1 Funcionalidade das operações

O sistema que foi desenvolvido neste trabalho deve ser capaz de cumprir todas as operações a que se propôs e para testar isso decidimos realizar uma série de experiências onde eram efetuadas várias operações pelos vários dispositivos móveis que faziam parte do sistema. Nessas experiências foram efetuadas operações de: publicação; subscrição no passado; subscrição no futuro; subscrição no passado e no futuro; *despublicação*; *dessubscrição*; após receber um notificação: obtenção das fotografias de imediato ou após a fotografia ter sido colocada para obtenção posterior.

***Churn*.** Além das situações referidas anteriormente, o cenário apresentado na Figura 5.4 permitiu avaliar o funcionamento das operações de cenários onde existe saída de nós ou a falhas de comunicação. No cenário experimental, o publicador que efetua as publicações sai da rede, e após a sua saída, um subscritor subscreve e pretende obter esses dados. Neste caso, o nó que deixou o sistema foi escolhido aleatoriamente, pois trata-se do publicador e este é escolhido aleatoriamente entre os nós disponíveis. Nas experiências realizadas, essas operações foram sempre concluídas com sucesso, comprovando a persistência dos dados no sistema. É de notar que a garantia do funcionamento da operação de obtenção neste caso é garantida às custas de um aumento na latência da operação, isto porque, se for selecionado o nó que saiu da rede ou se a mensagem se perdeu, é necessário repetir o pedido desta operação, como se pode ler na Secção 3.4.3, levando a uma aumento da latência.

Após a realização destas experiências ficou comprovado que o sistema cumpre com sucesso todas as operações que lhe são pedidas mesmo quando um publicador sai abruptamente da rede, obrigando a utilização de outra réplica desse objeto no sistema.

5.2.1.2 Latência das operações

A latência da execução das operações é uma medida importante para avaliar a usabilidade da aplicação e, neste caso, também é útil para experiênciar a usabilidade que a implementação do Thyme oferece às aplicações que o utilizam. A medição da latência das operações foi efetuada para as operações de publicação, *despublicação*, subscrição, *dessubscrição* e obtenção dos dados. É de notar que estas experiências foram efetuadas utilizando uma fotografia muito pequena, com 35 bytes, para ser possível mediar apenas a latência das operações com os dados gerados pelo sistema. A latência das operações de publicação e obtenção de dados poderão aumentar dependendo da fotografia em questão. Isto porque a latência da operação de publicação depende do tamanho da miniatura que será enviada – no nosso caso é do mesmo tamanho que a própria fotografia pois trata-se de um

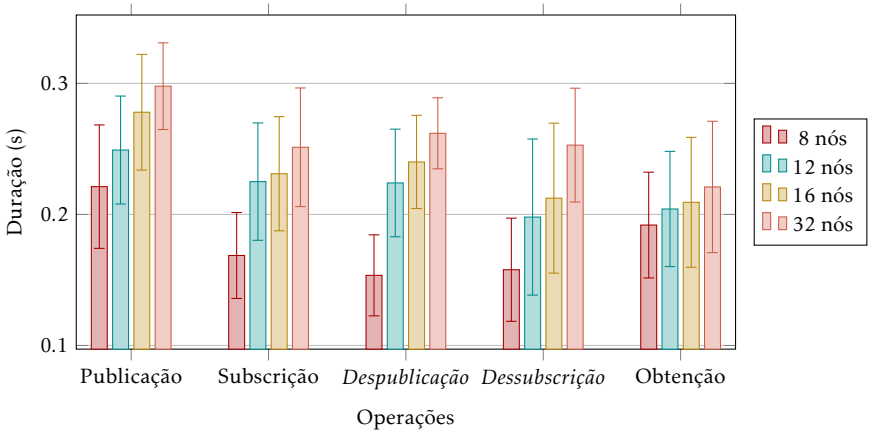


Figura 5.6: Latência das operações.

tamanho muito pequeno, mas normalmente a miniatura é relativamente mais pequena que a fotografia original – e a operação de obtenção depende da própria imagem que se pretende obter, sendo que um tamanho grande aumentará a latência da operação.

Na Figura 5.6 é possível verificar os resultados das experiências efetuadas, onde o número de dispositivos móveis foi variando.

A publicação é a operação que apresenta maior latência comparativamente à subscrição, *dessubscrição* e *despublicação*. Este situação deve-se ao facto de nesta operação serem tratados e enviados um conjunto de dados maior que as restantes operações – os metadados do objeto a partilhar (ver Secção 3.4.1). Um dos elementos dos metadados é uma miniatura da fotografia a partilhar, o que leva a que esta operação tenha mais trabalho, porque há a necessidade de gerar a miniatura, e que o envio dos metadados seja mais demorado, levando a um valor mais elevado a nível de latência.

Como referimos anteriormente, tanto a latência da publicação como a da obtenção dependem do tamanho da miniatura e do objeto a obter, respetivamente. Dessa forma, seria de esperar que a latência da obtenção se apresenta-se superior a da publicação, uma vez que na obtenção o objeto viaja na rede. Contudo, neste caso, como utilizamos uma imagem muito pequena – 35 bytes – o tamanho da miniatura igualou o tamanho da imagem a ser enviada. Logo, o valor da latência da publicação ser superior ao valor da latência obtenção justifica-se pelo maior número de informações enviadas e processadas na operação de publicação, os metadados, contrariamente a obtenção que apenas viaja na rede o objeto e o seu identificador. Numa situação com tamanhos de imagens superiores, a latência da obtenção será superior à da publicação, pois o tamanho da imagem será, certamente, maior que o tamanho da miniatura.

Em geral podemos considerar que todas as operações são executadas com tempos bastante aceitáveis, garantindo interatividade com o utilizador. Como esperado, o aumento do número de diapositivos na rede tem um pequeno impacto na latência das operações visto que existe mais tráfego na rede, o que leva a interferências na comunicação. Ainda assim, nenhuma das operações apresenta valores de latência superiores a 0.3 segundos,

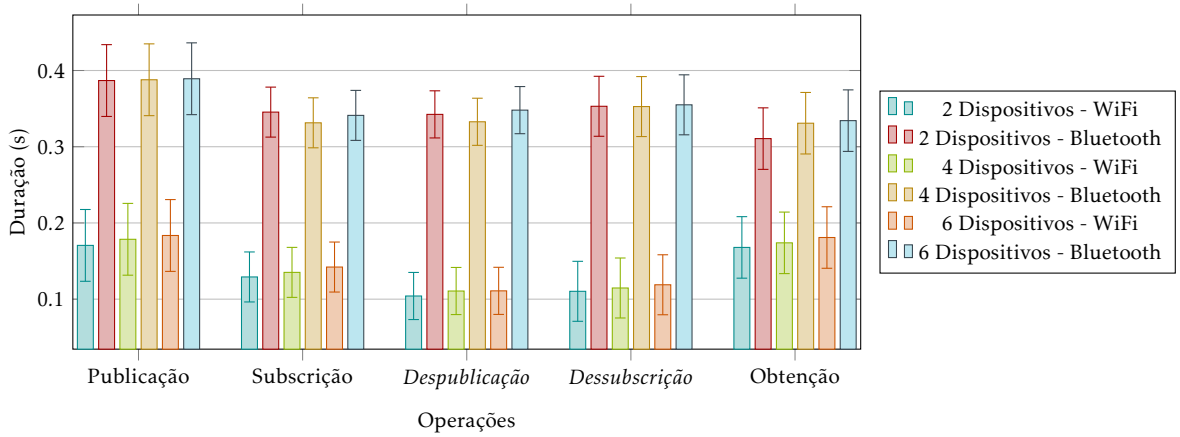


Figura 5.7: Comparação da latência das operações usando Wi-Fi e Bluetooth.

um valor perfeitamente aceitável para uma aplicação interativa, permitindo que todas as operações sejam executadas em tempo real, garantindo assim uma boa experiência de utilização para o utilizador.

Variando as tecnologias. Visto que a nossa implementação do THYME, e consequentemente a nossa aplicação, permite a utilização de várias tecnologias de comunicação para a criação da rede de dispositivos móveis, decidimos efetuar uma bateria de testes de forma a comparar a latência das operações efetuadas pelos utilizadores recorrendo a diferentes tecnologias de comunicação. Nestas experiências foram comparadas as latências das operações em redes utilizados o Bluetooth e um ponto acesso Wi-Fi. Devido a algumas limitações levantadas pela comunicação via Bluetooth apenas foi possível comparar os resultados com 2, 4 e 6 dispositivos simultaneamente. Os dados recolhidos nessas experiências encontram-se tratados e apresentados no gráfico da Figura 5.7.

Após a análise dos dados recolhidos, gráfico da Figura 5.7, é possível reconhecer facilmente que a comunicação através de uma rede formada utilizando Bluetooth é mais lenta quando comparada com a comunicação via Wi-Fi. Utilizando Wi-Fi, a latência das operações, na operação mais demorada, não ultrapassa os 0.2 segundos. Quanto a utilização de Bluetooth verificamos uma duplicação do valor obtido via Wi-Fi, isto é, cerca de 0.4 segundos na operação mais demorada. Apesar dos valores de latência utilizando Bluetooth serem relativamente mais elevados comparativamente ao Wi-Fi, pensamos que ainda são valores aceitáveis que garantem a interatividade do utilizador, permitindo oferecer ao utilizador uma experiência do seu agrado, e apresenta a vantagem de não necessitar de qualquer ponto de acesso.

Estes valores eram esperados, uma vez que a comunicação via Bluetooth apresenta uma taxa de transferência mais baixa comparativamente ao Wi-Fi.

Variando os tamanhos das fotografias. Como referido anteriormente, a latência da operação de obtenção de dados varia dependendo do tamanho dos dados que se pretendem

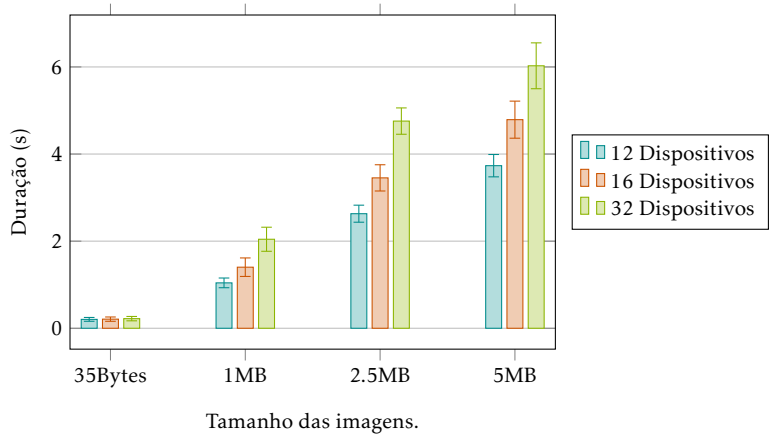


Figura 5.8: Comparação da latência da operação Obtenção, alterando o tamanho das imagens.

obter. De forma a medir o impacto do tamanho dos dados na latência desta operação, efetuamos experiências com imagens de 1MB, 2.5MB e 5MB, com os dispositivos móveis conectados a um ponto de acesso Wi-Fi. Os dados recolhidos encontram-se no gráfico da Figura 5.8.

Como seria de esperar, a latência da operação de obtenção de dados aumenta conforme o tamanho da imagem que se pretende obter. Mais uma vez, o aumento da latência conforme o aumento de dispositivos na rede está relacionado com as interferências, devido, não só mas também, ao aumento do tráfego na rede.

Apesar do crescimento dos valores da latência, através do gráfico verificamos que obter uma fotografia com 5MB numa rede com 32 dispositivos demora, em média, 6 segundos. Este valor parece-nos aceitável visto tratar-se de uma fotografia de alguma dimensão, permitindo ainda assim uma boa interação com o utilizador, garantindo-lhe pouco tempo de espera após efetuar esta operação. Não podemos esquecer ainda, que as operações são assíncronas, o que permite que o utilizador possa continuar a usufruir da aplicação durante o tempo de latência, garantindo a interatividade entre o utilizador e a aplicação.

5.2.1.3 Consumo energético

Quando estamos perante um contexto onde utilizamos dispositivos móveis, o consumo energético é um fator relevante que deve ser tido em conta. Estes dispositivos dispõem apenas da energia fornecida pela bateria intrínseca e que deve ser preservada para permitir o maior tempo de vida da mesma, não deixando o dispositivo inutilizável para o utilizador.

A implementação do Thyme desenvolvida foi pensada para poder ser utilizadas durante eventos de curta e média duração, logo devem apresentar consumos de energia baixos de forma a que seja possível a sua utilização durante os eventos referidos mas

também de forma a evitar o consumo de energia exaustivo, pois seria certamente do desagrado do utilizador.

De forma a avaliar o consumo energético do sistema implementado assim como da aplicação desenvolvida, realizamos experiências onde foram utilizadas todas as funcionalidades do THYME e onde foram medidos:

- O consumo de energia ao efetuar uma operação;
- O consumo de energia ao processar pedido de operação;
- O consumo de energia na manutenção do nó virtual, isto é, atualizar o estado após ser recebido um novo pedido para a célula.

Estas experiências, como referido anteriormente, foram efetuadas em redes de 8, 12, 16 e 32 dispositivos. Contudo, os dados referentes ao consumo de energia apenas foram recolhidos dos dispositivos Nexus 9, evitando assim variações nas medições que poderiam ocorrer se fossem utilizados dispositivos com características distintas, como é o caso dos testes com 32 dispositivos.

As medições de consumo de bateria foram efetuadas automaticamente, através de um módulo que utiliza a classe *BatteryManager* fornecida pelo Android.

Após uma análise dos resultados, concluiu-se que os valores de energia não dependiam do número de dispositivos na rede, pois verifica-se que a variação existente era reduzida e dessa forma optamos por apresentar os resultados destas experiências como uma média, independente do número de dispositivos na rede.

Consumo de energia ao efetuar operação. A Figura 5.9 apresenta os valores do consumo de energia de cada operação. A observação dos dados permite compreender que estamos perante consumos de energia baixos, visto que a operação mais custosa para o sistema não ultrapassa os 1,8 Joules. Contudo, esta representação não permite compreender os consumos energéticos em termos reais para o utilizador comum. De forma a apresentar os dados de forma mais "prática", é apresentado o gráfico da Figura 5.10.

O gráfico da Figura 5.10 apresenta os dados do consumo de energia da execução de cada operação durante 1 minuto. Durante 1 minuto foi simulado que o utilizador efetua sucessivamente cada operação logo consegue efetuar 60 operações. Temos noção que este seria um cenário exaustivo, logo será útil para testar o pior caso dos consumos de energia.

Os dispositivos em *standby*, isto é, apenas conectados ao ponto de acesso Wi-Fi (sem acesso à Internet), consomem cerca de 61 J/min. Ao executar a aplicação, o consumo de energia aumenta para cerca de 67 J/min, uma subida de 6 J/min, valor que achamos aceitável visto a apresentação do ecrã principal para o utilizador e o envio periódico de mensagens para os seus vizinhos. Estas mensagens – mensagens "Hello" (ver Secção 3.7) – são utilizadas para o controlo de elementos na célula e também para notificar possíveis novos elementos.

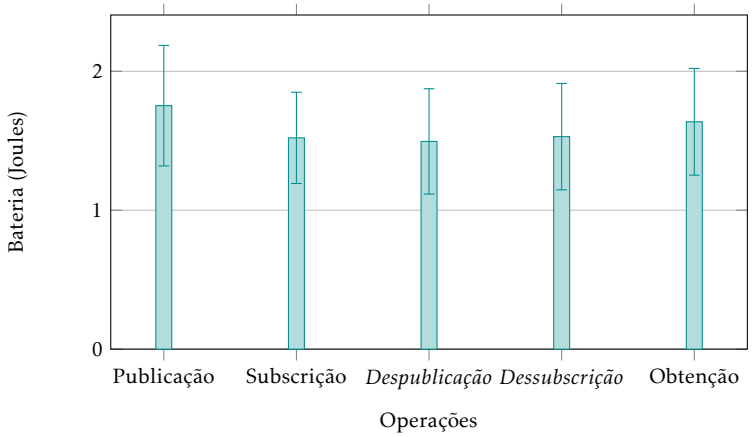


Figura 5.9: Consumos de energia ao efetuar pedidos.

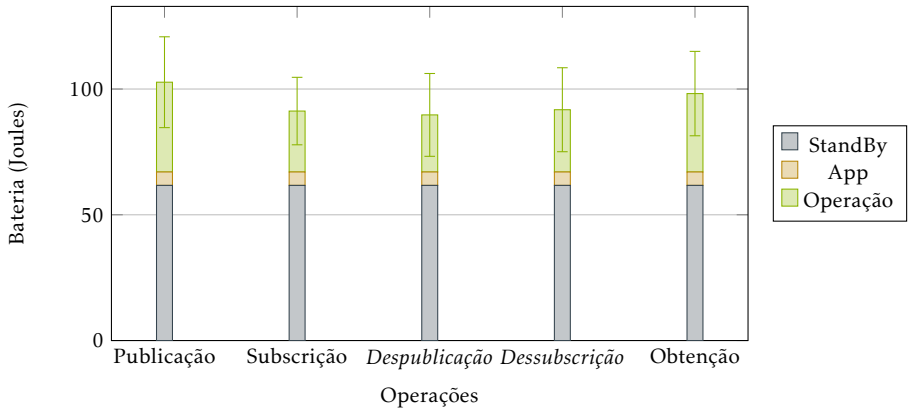


Figura 5.10: Consumos de energia ao efetuar pedidos durante 1 minuto.

Quando verificamos o consumo energético de cada operação, verificamos que apresentam valores muito próximos. Em média, a *despublicação* é a operação que consome menos recursos energéticos, consumindo perto de 90 J/min e no lugar oposto está a publicação que consome perto de 103 J/min de energia.

Dando uma perspetiva real, 1% da bateria dos dispositivos Nexus 9 corresponde a uma média de 960 Joules. De acordo com os dados apresentados, se o utilizador utilizar a aplicação intensivamente e de forma contínua durante 10 minutos, isto é, efetuar uma operação por segundo durante 10 minutos, consumirá em media 950 Joules, o que representará cerca de 1% da bateria do dispositivo.

Consumo de energia ao processar pedido. O nosso sistema trata-se de um sistema distribuído e por essa razão é comum serem outros nós a processar os pedidos quando um nó efetua esse pedido. Como seria de esperar, processar um pedido apresenta custos a nível de bateria para os dispositivos que efetuam esse trabalho. De forma a compreender o custo de processar um pedido de cada operação, é apresentado o gráfico da Figura 5.11.

Observando o gráfico da Figura 5.11 podemos verificar que também em relação ao

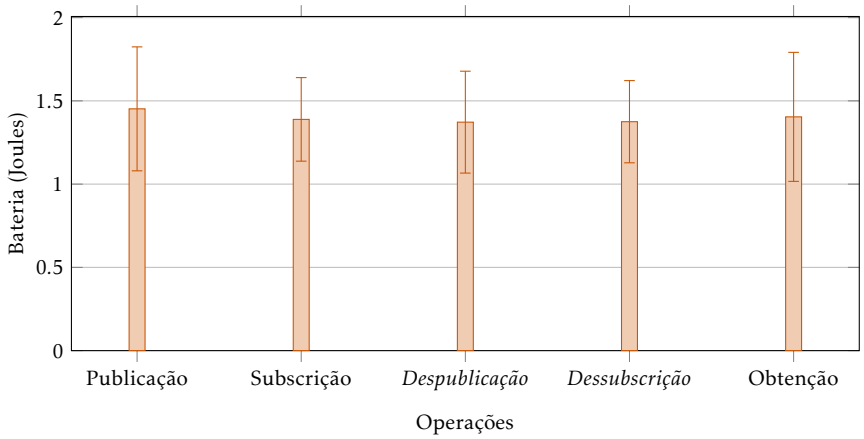


Figura 5.11: Consumos de energia ao processar pedidos.

processamento dos pedidos o consumo de energia é bastante semelhante em todas as operações. Em média, processar um pedido de uma operação custa cerca de 1.40J enquanto que efetuar um pedido custa perto de 1.60J, sendo uma diferença de 0.20J.

Num sistema que se insere no contexto em que estamos a trabalhar, é normal que todos os dispositivos do sistema trabalhem em prol do mesmo, por isso pensamos que este valor é aceitável tendo em conta que, provavelmente, não será sempre o mesmo nó a processar o pedido pois o trabalho é dividido por todos os elementos da célula.

Propondo uma situação limite, como a proposta em 5.2.1.3, simulando que um nó processa pedidos consecutivamente e intensamente durante 10 minuto, em média o nó irá gastar 830 Joules, o que significaria um consumo de perto de 0,85% do total da carga de um Nexus 9.

Consumo de energia na manutenção do nó virtual. A manutenção do conceito de nó virtual apresentado pela TDG-C também apresenta um acréscimo no consumo de energia nos dispositivos móveis. Quando existe um processamento de pedido de publicação, subscrição, *despublicação* ou *dessubscrição* por algum nó, os restantes terão de se atualizar com esse pedido e essa atualização consome recursos energéticos, como podemos ver no gráfico da Figura 5.12.

Através da análise do gráfico, é possivel verificar que os consumos de energias necessários para a manutenção do nó virtual, em cada operação, é muito semelhante ao custo de processar o próprio pedido, sendo um diferença, em média, de 0.1 Joules que se deve ao facto de não ser necessário o envio de mensagens no caso do processamento de replicação.

Seguindo a mesma linha, se o nó replicasse pedidos consecutivamente e intensamente durante 10 minutos, em média iria gastar 797 Joules, o que significaria um consumo de perto de 0,83% do total da carga de um Nexus 9.

Com estes resultados chegamos à conclusão que este é um ponto onde o nosso sistema pode ser otimizado. Visto que o consumo da manutenção do nó virtual é semelhante ao custo de processar um pedido, será um desafio tentar reduzir este custo de forma a

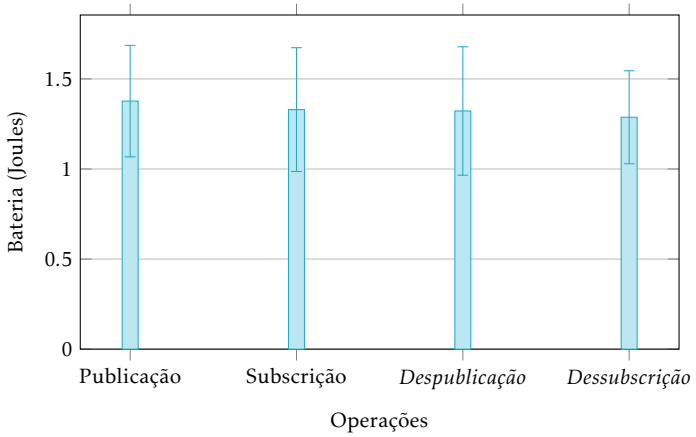


Figura 5.12: Consumos de energia na manutenção do nó virtual.

reduzir a energia gasta pelos dispositivos em trabalho realizado em prol do sistema.

Resumo. Resumindo, podemos considerar que a implementação do THYME que desenvolvemos, assim como a aplicação que utilizamos como caso de estudo, apresentam valores de consumos de energia bastante aceitáveis.

Tendo em conta que os dispositivos móveis podem efetuar pedidos, processar pedidos e também receber a replicação dos pedidos, no limite, um nó pode efetuar, processar e replicar pedidos ao mesmo tempo. Seguindo a situação limite referida anteriormente, supomos que o dispositivo efetua, processa e recebe replicações de pedidos consecutivamente e intensamente durante 10 minutos. No final dos 10 minutos o consumo de energia de todo este trabalho será de

$$950 + 830 + 797 = 2577 Joules,$$

correspondendo a menos de 3% da carga total do dispositivo Nexus 9, sendo um valor bastante aceitável.

Os resultados obtidos e apresentados sustentam o nosso argumento, comprovando que o THYME pode ser utilizado durante eventos de curta e média duração sem que o sistema apresente consumos excessivos de energia, não correndo o risco de uma descarga total da bateria do dispositivo móvel, o que o tornaria inutilizável.

5.2.1.4 Escalabilidade

Como referido inicialmente, no início da Secção 5.2, as experiências que realizamos para o tratamento de resultados, em geral, foram efetuadas em redes de 8, 12, 16 e 32 dispositivos. Tendo sido também realizadas experiências com 2, 4 e 6 dispositivos quando foi necessário comparar as diferentes tecnologias de comunicação.

Com esta grande variedade de combinações de dispositivos, foi possível comprovar a escalabilidade da implementação que efetuamos do THYME em ambiente real. Se observarmos os gráficos da latência das operações (Figura 5.6), podemos verificar que o

impacto na usabilidade da escalabilidade dos nós, de 8 para 32 não é muito relevante, continuando a ser um valor bastante aceitável na ótica do utilizador. Logo, é possível argumentar que, usado o cenário 2 (Figura 5.3), o sistema é escalável até 279 operações numa rede com 32 dispositivos móveis, sendo o impacto na usabilidade pouco relevante.

Com estes resultados conseguimos confirmar que o sistema desenvolvido é escalável, comprovando essa escalabilidade em contexto real para o qual o sistema foi desenvolvido.

5.2.2 Discussão

Os resultados obtidos em ambiente real da implementação do Thyme em redes móveis assim como a aplicação desenvolvida corresponderam às nossas expectativas.

Pensamos que a utilização de 32 dispositivos reais para a execução de testes experimentais é um número sólido de dispositivos, o que nos permitiu comprovar o funcionamento do sistema numa rede relativamente grande e a obtenção de resultados satisfatórios.

No final da análise dos resultados experimentais comprovamos que a implementação em causa cumpre todas as tarefas que lhe são incumbidas, mesmo quando há saída abrupta de elementos na rede, falha de comunicação ou *churn*, desde que todas as células estejam populadas e não fiques despopuladas, sendo estas questões limitações do sistema desenvolvido, abordadas na Secção 3.10.7.

Os resultados mostraram também que o sistema apresenta tempos de resposta bastante aceitáveis para as operações disponibilizadas, o que garante uma utilização interativa para o utilizador, tornando-se uma experiência agradável e comprovando a usabilidade da aplicação desenvolvida assim com a implementação do Thyme efetuada. Ficou comprovado igualmente que o sistema é operacional utilizando diferentes tecnologias de comunicação, sendo que no Bluetooth a latência das operações é superior quando comparado com a comunicação utilizando Wi-Fi.

A terminar, verificamos o consumo energético das operações, ficando comprovado que o Thyme pode ser utilizado em eventos de curta e média duração, pois apresenta consumos de bateria baixos quer efetuado trabalho pelo cliente como efetuado trabalho em nome do sistema.

5.3 Ambiente Simulado

A avaliação do sistema utilizando um ambiente simulado, foi efetuada utilizando um módulo de camada de rede que permite a comunicação em ambientes com fios. Desta forma foi possível simular vários nós, formando um rede de dispositivos móveis, onde foram executadas as operações e os comportamentos pretendidas. As experiências efetuadas foram direcionadas para avaliar o comportamento do sistema em larga escala, obter a carga nos nós em prol do sistema, avaliar o comportamento do sistema com mobilidades e com entrada e saída de dispositivos (*churn*).

Estas experiências foram controladas através de ficheiros de comportamento que indicam qual o número de nós no sistema, o momento da sua entrada e as operações a serem efetuadas por qual nó. Os nós são lançados em momentos aleatórios – possibilitando avaliar o comportamento do sistema com a entrada de nós no sistema, e são emitidos comandos de operações de forma sucessiva com um intervalo de tempo, definido previamente, entre operações. Como seria de esperar, apenas os nós que já estão na rede podem receber comandos de operação. Os nós quando terminam as suas operações saem do sistema, o que permite avaliar o comportamento do sistema quando existe saída dos nós.

A mobilidade dos nós é simulada utilizando um ficheiro de mobilidade. Esse ficheiro de mobilidade é lido pelo simulador e contém a posição inicial do nó, o momento em que se deve mover e qual a sua nova posição.

Nestes testes foi utilizado um mundo com 4 células, sendo que os nós foram aleatoriamente colocados pelas células disponíveis, sendo que todas as células eram povoadas e após a mobilidade ou *churn* nunca ficariam vazias.

5.3.1 Resultados Experimentais

As experiências efetuadas permitiram obter recolher um conjunto de resultados e análises que apresentamos nesta secção. Os resultados experimentais que apresentamos nesta secção, dizem respeito à execução do sistema em larga escala, à carga nos nós em prol do sistema, à resistência do sistema à mobilidade dos nós e ao seu comportamento em situações de entrada e saída de dispositivos – *churn*.

5.3.1.1 Funcionalidade em larga escala

A execução de experiências utilizando um ambiente simulado, permite experenciar a utilização do sistema com um número de nós superior aos utilizados em ambiente real. Desta forma, efetuamos testes onde utilizamos entre 30 e 200 nós. Os cenários realizados consistiam numa série de operações de publicações, subscrições de várias *tags* no passado e no futuro, obtenção, *despublicação* e *dessubscrição*. Nestes testes foram utilizados textos como objetos a partilhar no sistema.

Os resultados dos testes efetuados comprovaram que o sistema é funcional com um número bastante elevado de nós, conseguindo que todas as operações terminassem com sucesso.

5.3.1.2 Carga nos nós em prol do sistema

O Thyme é um sistema distribuído onde todos os nós do sistema trabalham em prol do mesmo. Dessa forma os nós efetuam trabalho que não é originado da vontade do utilizador, mas sim da necessidade do sistema. Visto isto é importante verificar a quantidade de trabalho que os nós têm de efetuar em nome do sistema. Os dados recolhidos sobre

a carga nos nós foram obtidos através do número de mensagens que, em média, cada nó recebe, processa e envia de cada tipo de operação em prol do sistema.

Nos gráficos da figuras 5.13 e 5.14 são apresentados os resultados obtidos, em média, sobre o número de mensagens que cada nó recebe e processa no caso da publicação, subscrição, replicação ativa e nova replica (Figura 5.13) e o número de mensagens que o nó envia a favor do sistema no caso dos *acknowledgments* e notificações (Figura 5.14). O gráfico representa a existência de 16 nós por célula, pois foi utilizado os resultados obtidos utilizando 32 dispositivos divididos por duas células.

O tratamento das mensagens de publicação, subscrição, replicação ativa e registo de nova réplica, (assim como *despublicação* e *dessubscrição*), crescem linearmente de acordo com o número de operações efetuadas (representadas pela sobreposição no gráfico da Figura 5.13) e não dependem do número de nós por célula. Visto que todos os nós da célula têm de se atualizar após este pedido, não há variações dependentes do número de nós na rede, sendo sempre linear de acordo com o número de operações.

Quando há N operações de publicação, subscrição, *despublicação* ou *dessubscrição* no sistema, cada uma das operações é processada por um nó da célula responsável pela *tag* respetiva de cada operação, sendo que em média, cada nó processa N mensagens desse tipo: um do nós processa a mensagem original, isto é, vindo do nós que efetuou o pedido e posteriormente dissemina essa mensagem pelos restantes nós da sua célula, levando a que todos os nós dessa célula recebam uma mensagem por pedido.

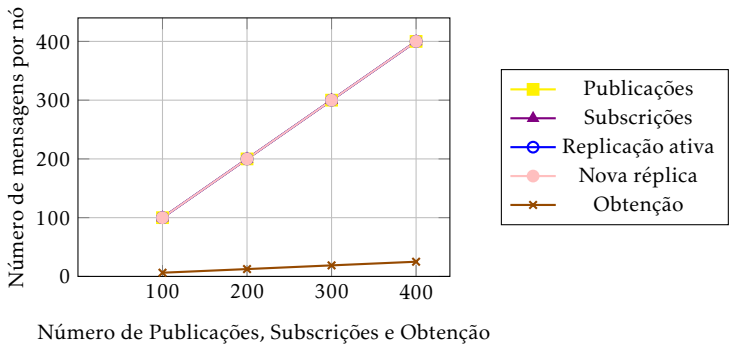


Figura 5.13: Número de mensagens recebidas por nó variando o número de operações.

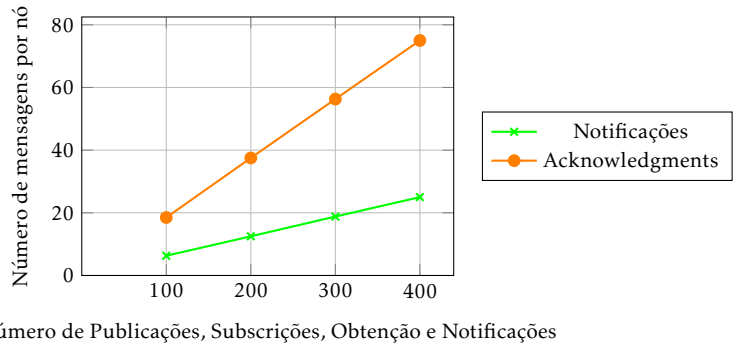


Figura 5.14: Número de mensagens enviadas por nó variando o número de operações.

Em relação à operação de obtenção de dados, neste caso fotografias, quando um nó efetua um pedido deste tipo, o pedido é recebido e processado por um nó e não existe replicação dessa mensagem pelos restantes nós. Dessa forma, o número de mensagens recebidas e processadas por cada nó aumenta conforme a subida do número de operações (linha com a cor castanha na Figura 5.13). Mas se a operação for efetuada com sucesso, é necessário informar a célula responsável pela *tag* que existe uma nova réplica desse objeto (linha cor de rosa no gráfico da figura 5.13). Neste caso é enviada uma mensagem para a célula respetiva com a informação que existe uma nova réplica de um objeto. Essa mensagem será processada por um nó e será transmitida pelos restantes nós, levando a que cada nó receba tantas mensagens quantas novas réplicas existentes.

O mesmo acontece em relação à replicação ativa, uma publicação obriga a uma disseminação dos dados publicados pelos seus vizinhos dentro da mesma célula, levando a que cada nó, em média, tenha de processar uma mensagem por publicação.

A carga do envio das mensagens de notificações e confirmação da operação (*acknowledgment*) dependem do número de nós por célula. Nestes casos, apenas um dos elementos da célula responsável efetua trabalho em prol do sistema, pois apenas o nó que processou o pedido envia a confirmação ou a notificação, dependendo dos casos, como podemos observar no gráfico da Figura 5.14. Neste gráfico podemos verificar que os nós precisam de enviar maior número de mensagens *acknowledgment* e notificação conforme a subida do número de operações. Com 100 operações de Publicação, Subscrição e Obtenção (300 operações no total) cada nó precisa de tratar e enviar, em media, 18.5 mensagens de *acknowledgment*, e quando passamos para 400 operações de cada um destes tipo fica cada nó responsável por cerca de 75 mensagens. Em relação as notificações o cenário é semelhante como podemos verificar acompanhado a linha verde do gráfico da Figura 5.14.

Nos gráficos da figura 5.16 e 5.15 podemos observar a carga de cada nó, em média, consoante o número de dispositivos na célula, quando efetuadas 300 operações (100 publicações, 100 subscrições e 100 obtenções) e 100 notificações respetivamente. Observando o gráfico, é possível concluir que quanto maior for o número de elementos na célula, menor é a carga em cada nó. É possível verificar que com 4 nós na célula, são enviadas 75 mensagens de *acknowledgment*, sendo reduzida essa carga para metade – cerca de 37.5 – quando a mesma experiência é efetuada com 8 elementos a popular a célula responsável pela *tag*.

Em resumo, podemos concluir que a carga de trabalho em nome do sistema nos nós das células responsáveis pelas *tags* apresenta uma tendência linearmente crescente de acordo com as operações efetuadas sobre essa *tag*. Esta carga deve-se à necessidade de manutenção do estado dentro da célula que é fundamental para o correto funcionamento do Thyme. Esta carga pode ser reduzida utilizando técnicas de replicação parcial, onde apenas alguns nós replicam os dados podendo ser esse número definido pelo sistema ou variando de acordo com a carga na célula. Este é um assunto que pode ser tratado posterioridade no seguimento deste trabalho.

A nível de notificações e *acknowledgment*, o incremento do números de nós na célula leva a uma redução da carga em cada nó.

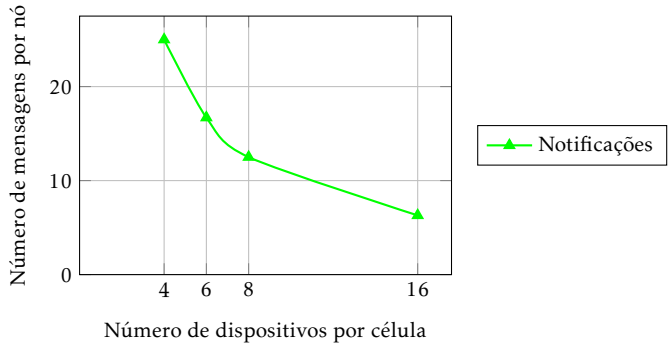


Figura 5.15: Número de mensagens por nó para 100 notificações, variando o número de nós.

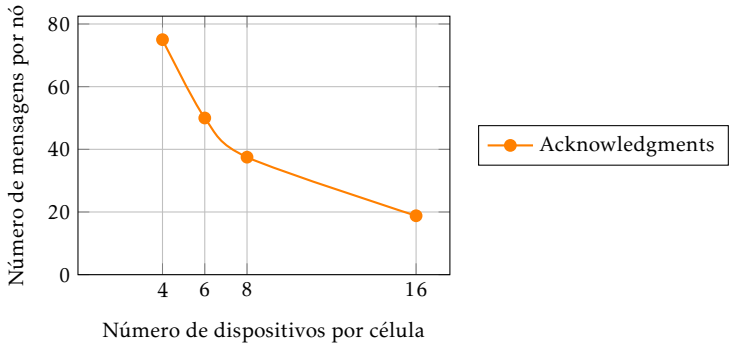


Figura 5.16: Número de mensagens por nó para 300 operações (100 publicações, 100 subscrições e 100 obtenções), variando o número de nós.

5.3.1.3 Carga com mobilidade

A mobilidade dos nós é uma característica inerente aos dispositivos móveis e por isso o sistema deve ser capaz de lidar com este tipo de situação.

Com vista a resolver este problema, é necessário efetuar uma série de comunicações para garantir o correto funcionamento do sistema, como abordado na Secção 3.9. Dessa forma, foi realizado uma bateria de experiências de forma a obter a quantidade de carga que o nó que se movimenta e os restantes nós dos sistema necessitam de suportar para permitir esse movimento.

Quando um nó se move, tanto os nós da nova célula como os nós da célula antiga precisam de efetuar trabalho devido a este movimento. Também as células responsáveis pelas *tags* das subscrições e dos objetos publicados pelo nó que se movimentou, necessitarão de efetuar trabalho em prol do sistema, isto se existirem.

Após parar, um dos nós da célula de destino terá de atualizar o novo nó na célula com o estado atual da célula.

Também é necessário atualizar a sua subscrição, na célula responsável por cada *tag* que subscreveu, e a localização de cada replica passiva, por cada *tag* dos objetos que obteve. Dessa forma, para atualizar a localização das réplicas passivas, envia uma mensagem por cada *tag* da replica, e para atualizar a subscrição, envia uma mensagem. O número de trabalho de cada nó, em cada célula responsável por cada *tag* subscrita, varia de acordo com o número de réplicas passivas correspondentes a essa *tag*, pois todos os nós da célula tem de atualizar o estado.

Em relação à célula antiga, cada nó terá de processar uma mensagem, a quando do movimento do nó, informando que o nó emissor se está a mover.

Ainda, se o nó tiver efetuado publicações no sistema, nesse caso é necessário atualizar os metadados dos dados publicados, indicando que se encontra numa célula diferente da célula onde foi publicado o objeto, passando o nó publicador a ser considerado uma replica passiva. Para isso, é enviada uma mensagem para as células responsáveis pelas *tags* dos objetos que publicou, sendo que todos os nós dessa célula têm de receber essa informação.

Após os resultados, chegamos à conclusão que o número de mensagens enviadas pelo nó que efetua o movimento é:

- No caso de ser publicador:

$$NrMensagensEnviadas = S + T_{RP} + T_P + 2$$

- No caso de não ser publicador:

$$NrMensagensEnviadas = S + T_{RP} + 2$$

onde, S**:** número de subscrições; T_{RP}**:** número de *tags* das réplicas passivas; T_P**:** número de *tags* dos objetos publicados; 2**:** 1 mensagem de aviso de movimento + 1 mensagem para atualização do estado.

No final desta análise, podemos concluir que o número de mensagens enviadas para resolver a questão do movimento varia com o número de subscrições, o número de objetos obtidos – pois tratam-se de réplicas passivas – e com o número de dados publicados pelo próprio nó.

Apesar de achar-mos que o trabalho efetuado compensa pelo resultado obtido, temos noção que este número de mensagens podia ser reduzido se fossem agregadas as mensagens para a mesma célula, como por exemplo a mensagem de atualização da subscrição com as réplicas passivas dessa *tag*. Neste caso poderíamos reduzir o número de mensagens, mas o trabalho no nó seria idêntico. Poderá ser uma melhoria a efetuar ao sistema no futuro.

5.3.1.4 Funcionalidade com movimento e entrada e saída de dispositivos - *Churn*

O Thyme deve permitir que todas as operações efetuadas pelo utilizador sejam terminadas com sucesso, mesmo que exista movimento dos nós ou *churn*. As experiências efetuadas com o simulador, comprovaram que as mensagens de publicação, subscrição, *despublicação* e *dessubscrição* e obtenção são concluídas 100% das vezes que são executadas, desde que sejam mantidas as condições adequadas de comunicação e cumpridas as limitações referidas na Secção 3.10.7.

A garantia de sucesso das operações de publicação, subscrição, *despublicação* e *dessubscrição* é obtida pelo mecanismo de reencaminhamento do módulo de Encaminhamento que garante que a mensagem é enviada para um elemento da célula respetiva. Se o elemento escolhido pelo mecanismo não estiver disponível, o Encaminhamento selecionará outro elemento da célula, garantindo assim que a operação é executada com sucesso.

No caso da obtenção, como pode ser selecionado uma réplica passiva, esse nó pode estar indisponível, devido a movimento ou *churn*, e neste caso é necessário escolher uma nova réplica, uma replica passiva ou optar por comunicar com a célula do publicador e optar por uma réplica ativa.

Tendo em conta as soluções implementadas para a resistência à mobilidade ou *churn*, com as experiências efetuadas foi possível verificar que se a taxa de *churn* ou de movimento for elevada pode levar ao aumento da latência das operações de forma a possibilitar a sua execução com sucesso, pois é necessário repeti-las no caso da obtenção, uma vez que é necessário escolher outra réplica, ou reenviar a mensagem para outro elemento da célula no caso das restantes operações.

Número de mensagens. Como referido, quando uma operação não tem sucesso é efetuado um reenvio do pedido até que a mesma seja concluída ou repetir a obtenção selecionando uma nova réplica no caso da obtenção. Como é necessário o reenvio das mensagens, o número de mensagens obrigatoriamente aumenta, levando ao aumento da latência, e por essa razão decidimos estudar essa situação.

No gráfico da Figura 5.17 estão representadas o número de mensagens que são enviadas, em média, por um nó quando este deseja efetuar 10 operações de obtenção, variando a taxa de *churn*/movimento dos nós, onde existem 10 réplicas de cada fotografia. Os nós vão saindo/movimentando-se da rede de forma aleatória.

Como seria de esperar, o número de mensagens enviadas aumenta com a percentagem de nós que se movem ou saem da rede. Ao analisar o gráfico da Figura 5.17 é possível observar que com 30% de nós indisponíveis é necessário enviar cerca de 15 mensagens para garantir o sucesso das 10 operações pretendidas.

Concluído, podemos afirmar que o Thyme garante o sucesso das suas operações, sendo resistente à mobilidade e *churn*, sendo necessário um aumento da latência dessas mesmas operações, devido a necessidade do envio de mais mensagens, dependendo da percentagem de nós indisponíveis na rede.

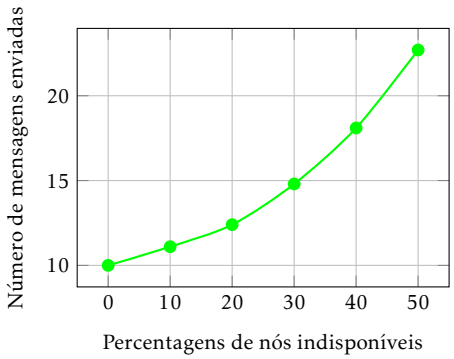


Figura 5.17: Número de mensagens enviadas variando a taxa de nós indisponíveis.

5.3.2 Discussão

Utilizando um ambiente simulado foi possível avaliar e comprovar algumas das funcionalidades e características que não foram experimentadas em ambiente real. No final desta secção, podemos afirmar que a implementação do THYME em redes de dispositivos móveis é escalável para um número elevando de nós, podendo ser utilizado em eventos que nos propomos como festas de anos ou reuniões.

Em relação à carga de trabalho dos nós em nome do sistema, pode-se concluir que dentro da célula responsável pela *tag*, éo trabalho é linearmente crescente de acordo com as operações efetuadas, e que o número de mensagens de notificações e *acknowledgment* diminui com o aumento da densidade dos nós dentro da célula.

Através da avaliação simulada também verificamos a capacidade de lidar com *churn* e com mobilidade por parte do THYME. O sistema consegue efetuar as operações que lhe são pedidas pelo utilizador mesmo com as características de mobilidade deste contexto, contudo existe um trabalho extra por parte dos nós que fazem parte do sistema que pensamos ser compensatório para o resultado final.

5.4 Sumário

O Capítulo 5 foi dedicado à explicação da avaliação efetuada ao sistema, aos resultados experimentais dessa avaliação e às conclusões e discussões originadas por esses resultados. As experiências realizadas foram efetuadas em dois ambientes distintos, um ambiente real e um ambiente simulado, o que permitiu uma avaliação a vários pontos do sistema e da aplicação desenvolvida como caso de estudo.

Nesta avaliação foi possível estudar a operacionalidade das funcionalidades do sistema, a latência dessas operações, o custo energético que apresentavam para o dispositivos móvel, o comportamento do sistema a cenários de mobilidade e *churn* e ainda a carga que os nós apresentavam para realizar trabalho em prol do sistema.

Em seguida, o último capítulo (Capítulo 6), apresenta as conclusões gerais desta dissertação e também alguns pontos que podem ingressar em trabalho futuro, quer a nível

de melhoria do sistema atual quer a nível de novas características.

Capítulo 6

Conclusão

Neste capítulo são apresentadas algumas conclusões referentes a esta dissertação e algumas orientações interessantes para desenvolver em trabalho futuro.

6.1 Conclusão

Nesta dissertação foi apresentada uma materialização do modelo Thyme [78] para redes de dispositivos móveis Android. Este sistema foi desenvolvido para colmatar as lacunas que existem na partilha e armazenamento de dados em situações onde os utilizadores se encontram geograficamente próximos, como festas de anos ou reuniões, e onde são gerados grandes quantidades de dados. A solução comum nesta situação seria a utilização de um serviço de *cloud* centralizado para a partilha dos dados, mas esta solução necessita de ligação à Internet, colocando grande carga na rede, levando a latências elevadas e até a falhas na rede. O Thyme surge como alternativa, utilizando os próprios dispositivos móveis para permitir a disseminação e armazenamento dos dados na periferias, mais próximo do utilizador sem necessidade de recorrer a serviços centralizados.

O Thyme permite efetuar as operações típicas de um sistema publicador/subscritor – publicação, subscrição e *dessubscrição* – mas também permite efetuar *despublicação* dos dados publicados uma vez que o sistema armazena os dados publicados de forma persistente. Além da *despublicação*, a subscrição é um pouco diferente do modelo publicador/subscritor comum, uma vez que permite ao subscritor subscrever o tópico do seu interesse, no presente, no futuro e no passado, sendo que a subscrição será ativa dentro do intervalo de tempo selecionado pelo utilizador. Como seria de esperar, o Thyme permite a obtenção dos dados notificados ao utilizador.

A abordagem escolhida para esta implementação do Thyme utiliza uma estratégia

de encaminhamento e armazenamento baseada numa tabela de dispersão geográfica baseada em células, onde o armazenamento é centrado em dados. Nesta abordagem o espaço geográfico onde o sistema está disponível é divido em células – quadrados de tamanho igual – onde todos os nós dentro dessa célula, colaboram entre si, formando um nó virtual. Estes nós virtuais são escolhidos, através da tabela de dispersão geográfica, como responsáveis por armazenar as publicações – mas não os dados publicados – as subscrições e verificar se publicações combinam com alguma das subscrições, ou vice-versa. Os dados publicados por sua vez ficam armazenados dentro da célula do publicador de forma a diminuir o tráfego na rede.

A implementação do Thyme num ambiente real, utilizando um rede de dispositivos móveis, levou à necessidade de tomar algumas decisões, assim como fazer algumas adaptações ao contexto. Neste contexto, é necessário que o utilizador defina qual o espaço geográfico onde o sistema é ativo, criando um "Mundo do Thyme", que será posteriormente dividido em células de forma a possibilitar o funcionamento da tabela de dispersão geográfica baseada em células. Utilizando dispositivos móveis, foi necessário obter a localização dos dispositivos, neste caso utilizando GPS, e detetar o seu movimento, se for caso disso, utilizando o acelerómetro, neste implementação.

Visto que o Thyme pode ser utilizado nos mais diversos casos de uso, foi necessário tornar possível a existência de várias instâncias do Thyme, sendo possível interagir entre as várias instâncias sendo que todas elas partilham a mesma componente de servidor mas cada uma possui uma componente de cliente única. Isto permite que cada instância seja capas de fazer as suas publicações, subscrições e obter dados, independentemente das restantes, mas o armazenamento dos dados é comum a todas as instâncias.

O caso de estudo desenvolvido para o Thyme foi a "Galeria de Fotos do Thyme". Esta aplicação para dispositivos móveis Android, permite a partilha de fotografias entre utilizadores geograficamente próximos conectados à mesma galeria, isto é, ao mesmo "Mundo do Thyme". Nesta aplicação o utilizador pode publicar as fotografias do seu dispositivo móvel associando-lhes uma ou várias *tags*, pode *despublicar* as fotografias partilhadas, pode subscrever *tags* e pode *dessubscrever* as *tags* subscritas. Após *subscrição* receberá notificação de fotografias que existam no sistema, correspondentes à *tag* subscrita e dentro do intervalo de vida da sua subscrição. Quando o utilizador for notificação, pode obter a fotografia notificada, pode decidir tomar a decisão mais tarde ou pode ignorar a fotografia.

A avaliação, em ambiente real e ambiente simulado, que foi efetuada ao sistema e à aplicação de caso de estudo desenvolvida permitiu comprovar que o sistema cumpre as operações oferecidas, mesmo com a existência de *churn* e movimento dos nós, desde que sejam respeitadas limitações do sistema. Isto porque a implementação do Thyme apresenta algumas limitações relacionadas com as células vazias.

A nível de usabilidade, as nossas experiências comprovaram que o sistema garante uma utilização interativa para o utilizador, tornando-se uma experiência agradável, devido aos valores baixos da latência das operações.

Comprovou-se ainda que o sistema é operacional utilizando diferentes tecnologias de comunicação, sendo testado o comportamento com Bluetooth e Wi-Fi, sendo confirmado que a latências das operações é superior utilizando a comunicação via Bluetooth.

A nível energético, ficou comprovado que o Thyme pode ser utilizado em eventos de curta e média duração, pois apresenta consumos de bateria baixos, tanto na execução das tarefas indicadas pelo utilizador, como a efetuar trabalho em nome do sistema.

Foi também estudada a carga que era colocada sobre os nós quando estes efetuavam trabalho em prol do sistema, quer para manter o estado do nó virtual, para responder aos pedidos de outros utilizadores ou para garantir que o sistema seja resiste ao movimento e *churn* dos nós. Apesar de ficar comprovado que manter o estado do nó virtual traz alguma carga aos nós, o nível de carga é aceitável visto que permite que os dados sejam armazenados de forma persistente no sistema.

Em conclusão, pensamos que os objetivos desta dissertação foram atingidos, com o desenvolvimento de uma materialização do Thyme capaz de garantir a partilha e o armazenamento de dados na periferia, utilizando uma rede de dispositivos móveis. A aplicação que desenvolvemos como caso de estudo permitiu-nos avaliar o sistema em ambiente real, de forma a comprovar que o sistema cumpre as nossas expectativas perante o contexto para que foi desenvolvido.

6.2 Trabalho Futuro

Atualmente, temos uma sistema funcional que está a ser utilizado como base de uma aplicação de partilha de fotografias na periferia. No entanto, existem ainda alguns problemas em aberto e questões que não fazendo parte do âmbito desta dissertação podem melhorar o sistema. Algum do trabalho que pode ser desenvolvido de forma a melhorar o sistema no futuro consiste em:

- mitigar os problemas relacionados com as células vazias, eliminando as limitações atuais do sistema, permitindo a mobilidade total dos nós. Ao longo do documento foram referidas algumas soluções para este problema que pode ser implementadas no futuro;
- resolver os problemas da função de distribuição para com as células vazias, implementado a solução previamente apresentada na Secção 3.10.7.
- utilizar uma função de dispersão mais sofisticada que consiga lidar com uma rede formada por múltiplos pontos de acesso Wi-Fi;
- implementar mecanismos de segurança, como controlo de acessos, de forma a ser possível tornar cada "Mundo do Thyme" privado, garantindo privacidade e proteção dos dados. Este trabalho já se encontra em desenvolvimento por outra equipa do projeto;

- implementar mecanismos de replicação parcial de forma a diminuir a carga dos nós, causada pela replicação ativa;
- utilizar um sistema de localização mais eficiente, que permita obter a localização dos nós em espaços fechados e que forneça todas as coordenadas geográficas possíveis (latitude, longitude e altitude);
- um problema mais específico que pode originar um trabalho futuro será, lidar com cenários onde a grelha se move, como por exemplo num barco. Desta forma o encaminhamento geográfico terá de se combinar com algoritmos de encaminho clássicos para lidar com esta situação.
- utilizar um mecanismo de movimento mais eficiente que permita a deteção do movimento do utilizador de forma mais fiável e precisa.

Estes são alguns assuntos que podem ser abordados em trabalhos futuros para melhorar a implementação do Thyme desenvolvida neste trabalho mas também para melhorar o projeto "Hyrax: Crowd-Sourcing Mobile Devices to Develop Edge Clouds" [57] em geral.

Bibliografia

[1] D. P. Agrawal e Q.-A. Zeng. *Introduction to wireless and mobile systems*. Cengage Learning, 2015.

[2] W.-F. Alliance. *Wi-Fi*. 2017. URL: https://www.wi-fi.org/.

[3] Anmobi.Inc. *Xender, Faster mobile file transfer app*. 2014. URL: http://www.xender.com/.

[4] F. Araujo, L. Rodrigues, J. Kaiser, C. Liu e C. Mitidieri. "CHR: a distributed hash table for wireless ad hoc networks". Em: *25th IEEE international conference on distributed computing systems workshops*. IEEE. 2005, pp. 407–413.

[5] M. Armbrust, A. Fox, R. Griffith, A. D. Joseph, R. H. Katz, A. Konwinski, G. Lee, D. A. Patterson, A. Rabkin, I. Stoica e M. Zaharia. *Above the Clouds: A Berkeley View of Cloud Computing*. Rel. téc. UCB/EECS-2009-28. EECS Department, University of California, Berkeley, 2009. URL: http://www2.eecs.berkeley.edu/Pubs/TechRpts/2009/EECS-2009-28.html.

[6] M. Armbrust, A. Fox, R. Griffith, A. D. Joseph, R. H. Katz, A. Konwinski, G. Lee, D. A. Patterson, A. Rabkin, I. Stoica e M. Zaharia. "A view of cloud computing". Em: *Commun. ACM* 53.4 (2010), pp. 50–58. DOI: 10.1145/1721654.1721672. URL: http://doi.acm.org/10.1145/1721654.1721672.

[7] R. Baldoni e A. Virgillito. "Distributed event routing in publish/subscribe communication systems: a survey". Em: *DIS, Universita di Roma La Sapienza, Tech. Rep* 5 (2005).

[8] S. Basagni, M. Conti, S. Giordano e I. Stojmenovic. *Mobile ad hoc networking: the cutting edge directions*. Vol. 35. John Wiley & Sons, 2013.

[9] A. Boukerche, B. Turgut, N. Aydin, M. Z. Ahmad, L. Bölöni e D. Turgut. "Routing protocols in ad hoc networks: A survey". Em: *Computer networks* 55.13 (2011), pp. 3032–3080.

[10] Business Wire. *Wireless Industry Continues Efforts to Boost Networks in Preparation for Presidential Inauguration*. https://goo.gl/3DMIE7. Jan. de 2009.

[11] H. Carreiro. *Da nuvem para o nevoeiro: a ascensão do fog computing*. 2016. URL: http://www.itchannel.pt/news/opiniao/da-nuvem-para-o-nevoeiro-a-ascensao-do-fog-computing.

[12] A. Carroll e G. Heiser. "An analysis of power consumption in a smartphone". Em: (2010).

[13] Y. Chen e K. Schwan. "Opportunistic overlays: Efficient content delivery in mobile ad hoc networks". Em: *ACM/IFIP/USENIX International Conference on Distributed Systems Platforms and Open Distributed Processing*. Springer. 2005, pp. 354–374.

[14] K. Chintalapudi, A. Padmanabha Iyer e V. N. Padmanabhan. "Indoor localization without the pain". Em: *Proceedings of the sixteenth annual international conference on Mobile computing and networking*. ACM. 2010, pp. 173–184.

[15] M. Cilia, L. Fiege, C. Haul, A. Zeidler e A. P. Buchmann. "Looking into the past: enhancing mobile publish/subscribe middleware". Em: *Proceedings of the 2nd international workshop on Distributed event-based systems*. ACM. 2003, pp. 1–8.

[16] Cisco. *Cisco Visual Networking Index: Global Mobile Data Traffic Forecast Update, 2016–2021 White Paper*. 2017. URL: http://www.cisco.com/c/en/us/solutions/collateral/service-provider/visual-networking-index-vni/mobile-white-paper-c11-520862.html.

[17] P. Costa, C. Mascolo, M. Musolesi e G. P. Picco. "Socially-aware routing for publish-subscribe in delay-tolerant mobile ad hoc networks". Em: *IEEE Journal on Selected Areas in Communications* 26.5 (2008), pp. 748–760.

[18] G. F. Coulouris, J. Dollimore e T. Kindberg. *Distributed systems: concepts and design*. pearson education, 2005.

[19] G. Cugola, H Jacobsen et al. "Using publish/subscribe middleware for mobile systems". Em: *ACM SIGMOBILE Mobile Computing and Communications Review* 6.4 (2002), pp. 25–33.

[20] U. Drolia, R. Martins, J. Tan, A. Chheda, M. Sanghavi, R. Gandhi e P. Narasimhan. "The case for mobile edge-clouds". Em: *Ubiquitous Intelligence and Computing, 2013 IEEE 10th International Conference on and 10th International Conference on Autonomic and Trusted Computing (UIC/ATC)*. IEEE. 2013, pp. 209–215.

[21] U. Drolia, N. Mickulicz, R. Gandhi e P. Narasimhan. "Krowd: A key-value store for crowded venues". Em: *Proceedings of the 10th International Workshop on Mobility in the Evolving Internet Architecture*. ACM. 2015, pp. 20–25.

[22] DropBox. *DropBox*. 2017. URL: https://www.dropbox.com/.

[23] I. Ekata Systems. *Drop*. 2013. URL: https://play.google.com/store/apps/details?id=com.ekatasystems.apps.drop.

[24] J. Erman e K. Ramakrishnan. "Understanding the Super-sized Traffic of the Super Bowl". Em: *Proceedings of the 2013 Conference on Internet Measurement Conference*. IMC '13. Barcelona, Spain: ACM, 2013, pp. 353–360. ISBN: 978-1-4503-1953-9. DOI: 10.1145/2504730.2504770. URL: http://doi.acm.org/10.1145/2504730.2504770.

[25] C. Esposito, A. Castiglione, F. Palmieri, M. Ficco e K.-K. R. Choo. "A Publish/Subscribe Protocol for Event-Driven Communications in the Internet of Things". Em: *Dependable, Autonomic and Secure Computing, 14th Intl Conf on Pervasive Intelligence and Computing, 2nd Intl Conf on Big Data Intelligence and Computing and Cyber Science and Technology Congress (DASC/PiCom/DataCom/CyberSciTech), 2016 IEEE 14th Intl C*. IEEE. 2016, pp. 376–383.

[26] P. T. Eugster, P. A. Felber, R. Guerraoui e A.-M. Kermarrec. "The Many Faces of Publish/Subscribe". Em: *ACM Comput. Surv.* 35.2 (jun. de 2003), pp. 114–131. ISSN: 0360-0300. DOI: 10.1145/857076.857078. URL: http://doi.acm.org/10.1145/857076.857078.

[27] P. T. Eugster, P. A. Felber, R. Guerraoui e A.-M. Kermarrec. "The many faces of publish/subscribe". Em: *ACM computing surveys (CSUR)* 35.2 (2003), pp. 114–131.

[28] E. Fidler, H.-A. Jacobsen, G. Li e S. Mankovski. "The PADRES Distributed Publish/-Subscribe System." Em: *FIW*. 2005, pp. 12–30.

[29] N. Forum. *NFC Forum*. 2017. URL: https://nfc-forum.org/.

[30] D. Frey e G.-C. Roman. "Context-aware publish subscribe in mobile ad hoc networks". Em: *International Conference on Coordination Languages and Models*. Springer. 2007, pp. 37–55.

[31] R. Friedman e A. K. Shulman. "A density-driven publish subscribe service for mobile ad-hoc networks". Em: *Ad Hoc Networks* 11.1 (2013), pp. 522–540.

[32] FrostWire.com. *FrostWire, Cloud Downloader, BitTorrent Client and Media Player*. 2016. URL: http://www.frostwire.com/.

[33] P. Gardner-Stephen e S. Palaniswamy. "Serval mesh software-wifi multi model management". Em: *Proceedings of the 1st International Conference on Wireless Technologies for Humanitarian Relief*. ACM. 2011, pp. 71–77.

[34] Google. *Android*. 2014. URL: https://www.android.com/.

[35] Google. *Google Drive*. 2017. URL: https://www.google.com/intl/pt-PT_ALL/drive/.

[36] R. L. Grossman. "The Case for Cloud Computing". Em: *IT Professional* 11.2 (2009), pp. 23–27. DOI: 10.1109/MITP.2009.40. URL: http://dx.doi.org/10.1109/MITP.2009.40.

[37] Y. C. Hu, M. Patel, D. Sabella, N. Sprecher e V. Young. "Mobile edge computing – A key technology towards 5G". Em: *ETSI White Paper* 11 (2015).

[38] Y. C. Hu, M. Patel, D. Sabella, N. Sprecher e V. Young. "Mobile edge computing—A key technology towards 5G". Em: *ETSI White Paper* 11 (2015).

[39] Y. Huang e H. Garcia-Molina. "Publish/Subscribe in a Mobile Environment". Em: *Wireless Networks* 10.6 (2004), pp. 643–652. DOI: 10.1023/B:WINE.0000044025.64654.65. URL: http://dx.doi.org/10.1023/B:WINE.0000044025.64654.65.

[40] C. Intanagonwiwat et al. "Directed Diffusion: A Scalable and Robust Communication Paradigm for Sensor Networks". Em: *MobiCom*. Boston, Massachusetts, USA: ACM, 2000. ISBN: 1-58113-197-6.

[41] iVinnyApps. *QIKSHARE*. 2016. URL: http://www.qikshare.io/index.html.

[42] B. Karp e H.-T. Kung. "GPSR: Greedy perimeter stateless routing for wireless networks". Em: *Proceedings of the 6th annual international conference on Mobile computing and networking*. ACM. 2000, pp. 243–254.

[43] G. Kaur e M. M. Fuad. "An evaluation of protocol buffer". Em: *IEEE SoutheastCon 2010 (SoutheastCon), Proceedings of the*. IEEE. 2010, pp. 459–462.

[44] J. F. Kurose e K. W. Ross. *Computer networking: a top-down approach*. Addison Wesley, 2013.

[45] P. J. Leach, M. Mealling e R. Salz. "A universally unique identifier (uuid) urn namespace". Em: (2005).

[46] J. a. Leitão e L. Rodrigues. "Overnesia: A Resilient Overlay Network for Virtual Super-Peers". Em: *SRDS*. IEEE, 2014. ISBN: 978-1-4799-5584-8.

[47] G. Li, A. Cheung, S. Hou, S. Hu, V. Muthusamy, R. Sherafat, A. Wun, H.-A. Jacobsen e S. Manovski. "Historic data access in publish/subscribe". Em: *Proceedings of the 2007 inaugural international conference on Distributed event-based systems*. ACM. 2007, pp. 80–84.

[48] LiveQoS. *SuperBeam, SuperBeam file sharing*. 2014. URL: http://superbe.am/.

[49] I. M. Lombera, L. E. Moser, P. M. Melliar-Smith e Y.-T. Chuang. "Mobile ad-hoc search and retrieval in the iTrust over Wi-Fi Direct network". Em: *Proc. 9th Intl. Conference on Wireless and Mobile Communications*. 2013.

[50] T. H. Luan, L. Gao, Z. Li, Y. Xiang e L. Sun. "Fog Computing: Focusing on Mobile Users at the Edge". Em: *CoRR* abs/1502.01815 (2015). URL: http://arxiv.org/abs/1502.01815.

[51] J. Luo, J. Hubaux e P. T. Eugster. "PAN: providing reliable storage in mobile ad hoc networks with probabilistic quorum systems". Em: *Proceedings of the 4th ACM Interational Symposium on Mobile Ad Hoc Networking and Computing, MobiHoc 2003, Annapolis, Maryland, USA, June 1-3, 2003*. 2003, pp. 1–12. DOI: 10.1145/778415.778417. URL: http://doi.acm.org/10.1145/778415.778417.

[52] K. Maeda. "Performance evaluation of object serialization libraries in XML, JSON and binary formats". Em: *Digital Information and Communication Technology and it's Applications (DICTAP), 2012 Second International Conference on*. IEEE. 2012, pp. 177–182.

[53] D. Malkhi e M. Reiter. "Byzantine quorum systems". Em: *Distributed Computing* 11.4 (1998), pp. 203–213.

[54] D. Malkhi, M. K. Reiter, A. Wool e R. N. Wright. "Probabilistic Quorum Systems". Em: *Inf. Comput.* 170.2 (2001), pp. 184–206. DOI: 10.1006/inco.2001.3054. URL: http://dx.doi.org/10.1006/inco.2001.3054.

[55] M. Mauve, J. Widmer e H. Hartenstein. "A survey on position-based routing in mobile ad hoc networks". Em: *IEEE network* 15.6 (2001), pp. 30–39.

[56] R. Monteiro, J. Silva, J. Lourenço e H. Paulino. "Decentralized Storage for Networks of Hand-held Devices". Em: *proceedings of the 12th EAI International Conference on Mobile and Ubiquitous Systems: Computing, Networking and Services on 12th EAI International Conference on Mobile and Ubiquitous Systems: Computing, Networking and Services*. ICST (Institute for Computer Sciences, Social-Informatics e Telecommunications Engineering). 2015, pp. 299–300.

[57] P. Narasimhan e F. Silva. *Hyrax: Crowd-Sourcing mobile devices to develop edge cloud*. 2017. URL: http://hyrax.dcc.fc.up.pt.

[58] OnePlus. *OnePlus 5*. 2017. URL: https://oneplus.net/pt/5/specs.

[59] Open Garden, Inc. *MeshKit SDK*. https://goo.gl/MwFLLg. 2016.

[60] J. Paiva, J. Leitao e L. Rodrigues. "Rollerchain: A dht for efficient replication". Em: *NCA*. IEEE. 2013.

[61] R. K. Panta, R. Jana, F. Cheng, Y. R. Chen e V. A. Vaishampayan. "Phoenix: Storage Using an Autonomous Mobile Infrastructure". Em: *IEEE Trans. Parallel Distrib. Syst.* 24.9 (2013), pp. 1863–1873. DOI: 10.1109/TPDS.2013.84. URL: http://dx.doi.org/10.1109/TPDS.2013.84.

[62] T. Pongthawornkamol, K. Nahrstedt e G. Wang. "The analysis of publish/subscribe systems over mobile wireless ad hoc networks". Em: *Mobile and Ubiquitous Systems: Networking & Services, 2007. MobiQuitous 2007. Fourth Annual International Conference on*. IEEE. 2007, pp. 1–8.

[63] *Protocol Buffers*. 2017. URL: https://developers.google.com/protocol-buffers/.

[64] *Protocol Buffers*. 2017. URL: https://developers.google.com/protocol-buffers/docs/overview.

[65] *Protocol Buffers*. 2017. URL: https://developers.google.com/protocol-buffers/docs/proto.

[66] A. Rai, K. K. Chintalapudi, V. N. Padmanabhan e R. Sen. "Zee: Zero-effort crowdsourcing for indoor localization". Em: *Proceedings of the 18th annual international conference on Mobile computing and networking*. ACM. 2012, pp. 293–304.

[67] L. Rangaswamy. *MediaBowl: FileShare*. 2016. URL: https://play.google.com/store/apps/details?id=com.kshemateq.partyapp.mediabowlfree&hl=pt_PT.

[68] S. Ratnasamy, B. Karp, L. Yin, F. Yu, D. Estrin, R. Govindan e S. Shenker. "GHT: a geographic hash table for data-centric storage". Em: *Proceedings of the 1st ACM international workshop on Wireless sensor networks and applications*. ACM. 2002, pp. 78–87.

[69] S. Ratnasamy, B. Karp, S. Shenker, D. Estrin, R. Govindan, L. Yin e F. Yu. "Data-centric storage in sensornets with GHT, a geographic hash table". Em: *Mobile networks and applications* 8.4 (2003), pp. 427–442.

[70] G. F. Riley e T. R. Henderson. "The ns-3 Network Simulator". Em: *Modeling and Tools for Network Simulation*. Ed. por K. Wehrle, M. Güneş e J. Gross. Springer Berlin Heidelberg, 2010, pp. 15–34. ISBN: 978-3-642-12331-3. DOI: 10.1007/978-3-642-12331-3_2. URL: http://dx.doi.org/10.1007/978-3-642-12331-3_2.

[71] J. Rodrigues, E. R. B. Marques, L. Lopes e F. Silva. "Towards a Middleware for Mobile Edge-Cloud Applications". Em: •. submitted to MECC 2017.

[72] J. Rodrigues et al. "Benchmarking Wireless Protocols for Feasibility in Supporting Crowdsourced Mobile Computing". Em: *DAIS*. Springer, 2016.

[73] Samsung. *The most important feature in a mobile device*. 2015. URL: http://www.samsung.com/ae/discover/your-feed/the-most-important-feature-in-a-mobile-device/.

[74] S. K. Sarkar, T. Basavaraju e C Puttamadappa. *Ad hoc mobile wireless networks: principles, protocols and applications*. CRC Press, 2007.

[75] K. Seada e A. Helmy. "Rendezvous regions: a scalable architecture for service location and data-centric storage in large-scale wireless networks". Em: *IPDPS*. IEEE, 2004.

[76] B. SIG. *Bluetooth*. 2017. URL: https://www.bluetooth.com/.

[77] A. Silberschatz, P. B. Galvin e G. Gagne. *Operating System Concepts*. 8th. Wiley Publishing, 2008.

[78] J. A. Silva, H. Paulino, J. M. Lourenço, J. Leitão e N. Preguiça. "I Know What You Did Last Summer: Time-Aware Publish/Subscribe for Networks of Mobile Devices". Em: *CoRR* abs/1801.00297 (2017). URL: https://arxiv.org/abs/1801.00297.

[79] J. A. Silva, R. Monteiro, H. Paulino e J. M. Lourenço. "Ephemeral data storage for networks of hand-held devices". Em: *Proceedings of the International Symposium on Parallel and Distributed Processing with Applications. IEEE*. 2016.

[80] J. A. Silva, J. Leitão, N. M. Preguiça, J. M. Lourenço e H. Paulino. "Towards the Opportunistic Combination of Mobile Ad-hoc Networks with Infrastructure Access". Em: *MECC@Middleware*. ACM, 2016.

[81] A. Sumaray e S. K. Makki. "A comparison of data serialization formats for optimal efficiency on a mobile platform". Em: *Proceedings of the 6th international conference on ubiquitous information management and communication*. ACM. 2012, p. 48.

[82] A. Teófilo et al. "GOCRGO and GOGO: Two minimal Communication Topologies for WiFi-Direct Multi-group Networking". Em: *MobiQuitous*. 2017.

[83] D. G. Thaler e C. V. Ravishankar. "Using name-based mappings to increase hit rates". Em: *IEEE/ACM Transactions on Networking (TON)* 6.1 (1998), pp. 1–14.

[84] K. Thilakarathna, A. C. Viana, A. Seneviratne e H. Petander. "Mobile social networking through friend-to-friend opportunistic content dissemination". Em: *Proceedings of the fourteenth ACM international symposium on Mobile ad hoc networking and computing*. ACM. 2013, pp. 263–266.

[85] K. Thilakarathna, H. Petander, J. Mestre e A. Seneviratne. "Mobitribe: cost efficient distributed user generated content sharing on smartphones". Em: *IEEE Transactions on Mobile Computing* 13.9 (2014), pp. 2058–2070.

[86] J. Thomas, J. Robble e N. Modly. "Off Grid communications with Android". Em: *2012 IEEE Conference on Technologies for Homeland Security (HST)*. 2012.

Filipe Alexandre Bandeira Cerqueira

Licenciado em Engenharia Informática

Um Sistema Publicador/Subscritor com Persistência de Dados para Redes de Dispositivos Móveis

Dissertação para obtenção do Grau de Mestre em

Engenharia Informática

Setembro, 2017

FACULDADE DE
CIÊNCIAS E TECNOLOGIA
UNIVERSIDADE NOVA DE LISBOA

Filipe Alexandre Bandeira Cerqueira

Licenciado em Engenharia Informática

Um Sistema Publicador/Subscritor com Persistência de Dados para Redes de Dispositivos Móveis

Dissertação para obtenção do Grau de Mestre em

Engenharia Informática

Setembro, 2017

FACULDADE DE
CIÊNCIAS E TECNOLOGIA
UNIVERSIDADE NOVA DE LISBOA

Filipe Cerqueira 2017

2017 Um Sistema Publicador/Subscritor com Persistência de Dados para Redes de Dispositivos